Universidad
Carlos III de Madrid

TRABAJO DE FIN DE GRADO

OPTIMIZACIÓN DE CONSULTAS EN BASES DE DATOS RELACIONALES

Autora: Raquel García Frutos

Grado: Doble Grado en Ingeniería Informática y Administración de Empresas (2.010 - 2.016)

Tutora: Ana María Iglesias Maqueda

Entrega: 26 de Septiembre de 2016

Resumen ejecutivo [*Abstract*]

El propósito de este estudio ha sido proporcionar a alumnos universitarios y otros interesados en la rama del conocimiento de las bases de datos una visión completa del estado actual, haciendo énfasis en las bases de datos relacionales.

Con tal objetivo se ha dividido el estudio en dos grandes partes: una teórica y otra práctica. La primera lleva a cabo un análisis que parte de la rivalidad entre bases de datos relacionales y no relacionales para terminar centrándose en el estudio de uno de los sistemas gestores (Oracle XE 11g r2) y dos de las herramientas de gestión más extendidas, Oracle SQL Developer y TOAD. Esta primera parte incluye también una comparativa de los principales gestores de bases de datos y herramientas de gestión que justifiquen la elección final.

La segunda parte tiene una orientación eminentemente práctica, en la que se pretende completar el análisis de estos elementos utilizando para ello tres casos de prueba de distinta complejidad sometidos a diversas pruebas.

La sección final está dedicada a exponer las principales conclusiones del estudio, limitaciones bajo las cuales se ha desarrollado y propuestas de trabajos futuros que puedan utilizar el presente como base de partida.

Añadidamente, dado el enfoque académico adoptado, se ha incluido una última parte de anexos en los que aquellos interesados en ampliar información puedan consultar manuales de instalación de las herramientas empleadas, comandos de ejecución, visualizar los resultados a los que se hace referencia en el estudio, así como una propuesta para una guía de trabajo basada en TOAD.

A lo largo del estudio ha sido posible comprobar la extensa variedad de sistemas y herramientas hoy en día disponibles para manejar información almacenada en bases de datos, acrecentada por la veloz generación de nuevo conocimiento en esta área. La potencia de algunos de estos recursos tampoco pasa desapercibida, y la disposición de recursos como SQL Developer o TOAD hace posible trasladar parte de vasto ámbito al entorno académico.

Abstract

The purpose of this research was to provide university students and third party interested in the database knowledge a complete view of their current state, the available systems and tools related to them, focused mainly on relational databases.

In order to accomplish this task, the study has been conducted both from a theoretical and practical point of view. First of all, an analysis over relational and no relational databases fixes the base for the whole project, which will then be leaded to relational database management systems. Once a RDBMS is selected, which will be Oracle XE 11g r2, a variety of tools will be tested to find the two of them that better adjust to the project's necessities. Oracle SQL Developer and TOAD have been selected among other three alternatives. All these analysis include comparative tables that will help understand and summarize information.

Second phase of the project is dedicated to perform tests based on examples designed of different complexities that involve a wide variety of functions and subqueries, all of them defined into the HR schema provided by default in Oracle's Database XE 11g r2.

To conclude, a final section will summarize the most relevant conclusions, as well as the restrictions under which this study has been performed. We also provide a list f potential future works that may be attractive to conduct based on this one.

Additionally, due to the academic point of view that has lead all the project, there is a chapter in which several annexes have been included. They will help to install and deploy the tools that have been used, understand and visualize the optimal alternatives that were gathered during the tests phase, and a work guide proposal to be used by interested students that would like to learn the basics of TOAD and take advantage of this document.

Agradecimientos

Este trabajo no habría sido posible sin la guía y apoyo de mi tutora, Ana María Iglesias Maqueda.

A mis padres Susana y Ángel, mis hermanos Luis y Alejandro, y otros familiares cercanos que han estado siempre a mi lado apoyándome.

A todos vosotros, gracias de corazón.

Índice

Índice de Tablas

Índice de Ilustraciones

1. Introducción [*Introduction*]

Las bases de datos relacionales llevan desde su aparición en la segunda mitad del siglo XX ayudando a modelar el mundo que nos rodea, permitiendo extraer información a partir de los datos previamente recopilados. Un ejemplo son las bases de datos que las empresas desarrollan de sus empleados, con su identificador dentro de la empresa, sus datos personales, su puesto de trabajo y su responsable inmediato. Ya sea esta u otra base de datos, el fin es siempre el mismo: manejar y extraer información que aporte una utilidad a la empresa.

Detrás de la consulta más sencilla existen multitud de procesos que entran en acción para devolver la información requerida. El coste de estos procesos aumenta significativamente con el tamaño de la base de datos, tanto en tiempo necesitado como en consumo de CPU o el número de transacciones de tipo entrada - salida, además de un consumo mayor de energía eléctrica. Por ello, conocer el impacto de las operaciones ayuda a mejorar la definición de la consulta y optimizar sus costes.

A primera vista esto puede parecer algo rutinario, sin complicación y rápido. La realidad es que la evolución tecnológica es tal que la cantidad de conocimiento disponible acerca de nuevos sistemas gestores de bases de datos, lenguajes de consulta, técnicas de optimización, arquitecturas y demás componentes relacionados es inabarcable para una sola persona. Para superar este obstáculo, se han desarrollado herramientas que permiten a un usuario mejorar sus consultas, entre las cuales se encuentran SQL Developer y Toad.

Los cuatro objetivos básicos que pretende perseguir este estudio son:

1) Dar una visión global del estado actual de los sistemas gestores de bases de datos relacionales (RDBMS). Existe una gran variedad de la cual se pretende dar una pequeña muestra, de la variedad de lenguajes que pueden emplearse, de la compatibilidad entre ellas, de los sistemas operativos sobre los que pueden emplearse, y ahora hasta de soluciones en la nube. Todo ello con el fin de que el usuario sea consciente del elevado grado de adaptación que puede conseguir entre sus necesidades y las soluciones disponibles.

2) Aportar en un solo documento una revisión de los mecanismos disponibles para monitorizar el comportamiento de una sentencia SQL. Como se verá próximamente, existe un amplio abanico de mecanismos que permiten analizar el comportamiento interno de una consulta durante su ejecución que se ha querido plasmar en este proyecto con el fin de que el lector cuente con la existencia de estas herramientas para otros proyectos en el futuro y no se centre exclusivamente en las dos o tres más conocidas.

   Se ha decidido incluir las principales funciones del paquete dbms_xplan para analizar tanto los explain plan como los execution plan, la función Autotrace tanto en consola SQL*Plus como en SQL Developer, SQL Monitor, la recopilación de

estadísticas, los informes tkprof, el paquete dbms_profiler, el paquete dbms_result_cache, los hints, las variables de enlace o bind y algunas de las vistas más relevantes.

3) Comparar desde un punto de vista teórico-práctico la utilidad y aplicabilidad de herramientas y mecanismos disponibles para mejorar las consultas SQL. Por un lado se encuentran las aplicaciones teóricas de cada uno de los mecanismos y herramientas, y por otro el caso concreto para el que se quieran emplear (del cual dependen), por lo que se ha diseñado un entorno de pruebas bajo el cual se decidirá cuáles de estas facilidades pueden ser o no aplicadas y en qué grado, así como sus ventajas y limitaciones correspondientes. Junto con los mecanismos mencionados anteriormente, se han incluido además dos de las herramientas más conocidas: SQL Developer y Toad. En ambos casos se ha presentado una revisión teórica de la herramienta para a continuación evaluarla en función de los resultados obtenidos en el entorno de pruebas.

4) Sentar las bases para que el material pueda ser objeto de estudio en unas prácticas docentes de la Universidad. La elección de las herramientas y mecanismos tiene muy presente el usuario objetivo para el que está diseñado este estudio, lo que ha motivado la elección de algunas de las herramientas y el diseño de la base de datos empleada en el entorno de pruebas. El objetivo es proveer a los alumnos de un entorno de pruebas al que puedan acceder tanto en la Universidad como en casa, para que su uso no esté restringido exclusivamente a horario lectivo.

Para ello, se han diseñado cuatro grandes bloques de contenido. El primero abarca las secciones 1 y 2 en las que se presenta el proyecto y los aspectos más relacionados con la gestión de proyectos, centrándose especialmente en la metodología, el ciclo de vida, el análisis de costes y su oferta asociada, así como la planificación Gantt inicial y final.

Un segundo bloque en el que se incluyen las secciones 3 y 4 se encargan de cubrir los objetivos 1 y 2 del estudio, dando una visión completa del estado de las bases de datos relacionales en la actualidad utilizando para ello múltiples enfoques: otras alternativas con las que compite, el entorno socio-económico en el que están inmersas y las principales medidas legislativas utilizadas en su regulación. Una vez centrado el estudio en ellas, el siguiente paso está dedicado al análisis de los mecanismos y herramientas relacionadas con la mejora de las consultas.

La mitad de la sección 4 y la sección 5 completa cumplirán el tercer objetivo, por lo que se partirá de tres consultas iniciales en un entorno de pruebas definido que serán mejoradas utilizando dos vías principales: 1) SQL Developer y 2) Toad. Los resultados serán comparados y valorados en conjunto en la sección 5, dedicada además a mostrar las principales conclusiones del estudio, sus limitaciones y sus potenciales vías de desarrollo futuras.

El cuarto objetivo estará presente a lo largo de todo el trabajo, y especialmente en las partes dedicadas a explicar cómo crear el entorno de pruebas, instalar las herramientas, emplear los mecanismos e interpretar y valorar los resultados. Se han incluido con el mismo fin una sección de glosario y de webgrafía con recursos que serán de utilidad a los futuros lectores, así como una lista de los principales comandos que ha sido necesario utilizar durante el proyecto para activar los mecanismos utilizados en la monitorización de las consultas.

Introduction

Since their appearance in the 20th century, relational databases have been helping us in a variety of tasks related to model the world surrounding us and extracting information from these models. One of the simplest examples is the human resources database every company has when it reaches a certain number of employees. This database contains relevant information of each employee, like the employee id, their personal data, the job that it's been occupied and their direct manager. Independently of which database is it, the purpose is still the same: store data that can be used to retrieve information useful for the company.

Behind every single query there are multiple processes that are triggered in order to show the information required. Each of these processes has a cost, and it increases with the database size. Cost can be measured in terms of CPU usage, number of IO operations and even in terms of electric energy consumed. Knowing how a query demands all these resources will help optimize them, which will also have an impact in the whole system.

At first sight, this optimization process might seem quick and simple, but it is not. Not only because each query and the environment in which it is launched are unique, but also because the evolution of technology has been so fast that the amount of available knowledge related to databases, query languages, optimization techniques and new architectures is excessive only for one person. In order to solve this problem, several tools have been developed lately to help the user improve the queries and among them we find SQL Developer and Toad.

This project pursues four main objectives:

1) *Give a global view of the current variety of relational database management systems (RDBMS). Nowadays there is an extended buch of solutions of which this project intends to show a sample. These different solutions include several query languages, operative systems, compatibilities, and even cloud - supported, which proves the high adaptability the user may find between their own environment and the solutions available.*

2) *Review in only one document the main mechanisms and features currently available to monitor an SQL query. This paper pretends to show an overview of*

these tools so the reader can learn other methods to monitor the queries behavior once launched. Since there is an increasing number of monitoring tools, it is important for the users to keep updated and this paper tries to contribute to this task.

*Among the monitoring tools, this project includes the following: dbms_xplan Oracle package to analyze explain and execution plans, the autotrace function (used both from the SQL*Plus command line and SQL Developer), SQL Monitor, statistics gathering, tkprof reports, dbms_profiler package, dbms_result_cache package, hints, bind variables and some of the most important views.*

3) *Objective 3 consists on comparing from a theoretical and practical point of view the main tools used in this project to improve queries. It is not just about comparing tools, it also pretends to evaluate each of them depending on the features they offer to the users, to determine which of them is more recommendable in our test environment. In addition to the above mentioned mechanisms we include also two visual tools: SQL Developer and Toad. Each of them will be described in its own section and then will be compared using the obtained results.*

4) *Be the base for future academic exercises. One of the factors that were took into account during the tool selection process was how useful would it be for students, so they would be able to use them anytime and anyplace without needing to use only devices located in university.*

This document is divided into four main parts. The first one includes sections 1 and 2,it gives an overview of the project, its objectives and the principal software measures as methodology, life cycle, the planned costs and the hypothetical offer made to the client, as well as two Gantt graphics which show the initial and the final planning.

The second part is centered in objectives 1 and 2. Along sections 3 and 4 a complete view about relational databases and several mechanisms and tools is provided through multiple points of view: it shows different alternatives to the same relational database management requirements, the socioeconomic and legal environment in which these products are settled, and a specific analysis for each of them. In particular, subsection 4.1 and 4.2 will describe the tools previously mentioned, including SQL Developer and Toad.

Third objective is covered in sections 4 and 5. Based on three queries and the environment developed for testing purposes we will discuss the potential improvements applied to the queries through two main ways: 1) using Oracle's SQL Developer and 2) using Dell's Toad for Oracle. The results will be compared in section 5, which also includes conclusions, limits and proposed future extensions of the project.

There is not a particular section dedicated to the fourth objective. This is because the whole project is built upon it. As it was said before, the main point of view of this document is academic, so all the descriptions, analysis, explanations and results presentations try to increase the knowledge of the reader. Additionally, other sections have been included in the document to provide extra information that complete and help understanding the concepts covered along the paper. A glossary, a webgraphy with several online resources, two installation guides for SQL Developer and Toad, and a list of important commands that show how to use some of the functions used to monitor queries complete this paper.

2. Gestión del proyecto

Esta sección estará dedicada a presentar los aspectos de planificación del proyecto, divididos en tres subsecciones.

La primera se centra en la elección de la metodología y el ciclo de vida para este estudio, teniendo en cuenta sus características y diferencias respecto a un proyecto software habitual.

La segunda plasma el enfoque del primer punto en una planificación inicial del avance del proyecto mediante un gráfico Gantt, así como la propuesta inicial facilitada al cliente.

Por último, se muestra el avance real que ha tenido el proyecto y las justificaciones de las desviaciones más importantes.

2.1 Gestión del Software: Metodología, Ciclo de Vida

Se ha decidido abordar el proyecto utilizando una metodología ágil, inspirada en la diseñada por Craig Larman y basada en RUP (*Rational Unified Process*). Esta metodología se distingue por aplicar un método evolutivo (iterativo y adaptativo), incremental y dirigido por casos de uso, lo que significa que el estudio total se basa en los estudios progresivamente realizados (a los que también se conoce como 'rodajas').

La visión general del proceso está estructurado en torno a tres macroetapas: 1) Planificación, 2) Construcción, e 3) Instalación.

La primera etapa se centra en planificar el avance del estudio. Aunque en el método original se hable de extraer requisitos y construir prototipo porque el fin sea construir un software, en este estudio se incluyen las tareas de contacto con el cliente para extraer qué necesidades y expectativas tiene del estudio (ver sección 2.2), por lo que los requisitos no seguirán la misma clasificación.

La segunda etapa concentra la mayor parte de carga de trabajo. Se subdividirá en subetapas más cortas, basadas cada una en el feedback obtenido del cliente en la subetapa anterior. El contenido de cada etapa incluirá tareas de investigación, implementación, pruebas, documentación y control por parte del cliente, aunque según el nivel de avance del proyecto la carga de investigación e implementación podrá variar.

Por último, la etapa de instalación será en la cual se presente al cliente los resultados finales del estudio llevado a cabo, así como la respectiva documentación.

Como se ha indicado previamente, esta metodología es una adaptación de la original creada por Craig Larman, que está enfocada al desarrollo de software orientado

a objetos, por lo que hay fases que no se aplicarán al presente proyecto o se modificarán para adaptarlas.

El ciclo de vida que más se ajusta a este enfoque sería el cascada, en el que el orden simple de procesos es el siguiente: Requisitos > Diseño > Codificación > Pruebas > Operación.

Llevado a la metodología descrita, estos procesos se repetirían en cada una de las rodajas que incluye la segunda macroetapa, teniendo en cuenta que las primeras rodajas tendrán mayoritariamente carga de análisis e investigación (que ocuparían los puestos de requisitos y diseño), mientras que las centrales y últimas incorporarán progresivamente tareas de codificación y pruebas (de los casos de uso diseñados en los anteriores procesos).

2.2 Planificación Inicial: GANTT y Oferta

La planificación inicial del estudio estará formada por una oferta oficial propuesta al cliente y un gráfico GANTT con la estimación de tiempos iniciales.

El fin de la oferta es poner en conocimiento del cliente los objetivos que se pretenden alcanzar durante el proyecto, el enfoque que se adoptará y los medios y recursos que se estima que serán necesarios. Se incluye también una presentación del equipo asignado al estudio.

Este estudio pretende ofrecer una visión analítica de dos herramientas utilizadas para la gestión de las bases de datos relacionales y en concreto la optimización de consultas gracias a las funciones ofrecidas respectivamente. Su ámbito de aplicación es esencialmente académico, por lo que las revisiones de las alternativas consideradas y las conclusiones podrán incluirse como material de estudio.

El equipo encargado del proyecto está integrado por una persona, Raquel García Frutos. Las tareas de investigación, análisis, documentación y pruebas serán competencia de Raquel García. Por su parte Ana María Iglesias Maqueda, como cliente, se encargará de la aprobación final del estudio.

Los recursos requeridos para acometer el estudio se dividen en tres categorías: recursos hardware, recursos software y recursos de personal.

- Recursos hardware: Ordenador portátil con privilegios de administrador para instalar nuevas herramientas.

- Recursos software: Concentra la mayor parte de los recursos.
    - Oracle XE 11g r2: licencia gratuita.
    - SQL Developer: licencia gratuita, las funcionalidades avanzadas de pago no entran en el alcance del estudio.
    - TOAD: licencia temporal gratuita.
    - Paquete Microsoft Office: la documentación se generará mediante Microsoft Word, Microsoft Excel y MS Project, combinado con Microsoft Power Point en la presentación final de las conclusiones.
    - Sistema Operativo: Windows.
    - Sistema de backup: Google Drive y almacenamiento externo.
    - Notepad++: licencia gratuita. Utilidad: revisar informes generados cuyas extensiones son .txt, .trc, .prf, y .sql.
- Personal:
    - Raquel García Frutos

A continuación se presentan el cronograma de actividades y el gráfico Gantt correspondiente a la estimación inicial. La unidad temporal utilizada para medir el avance es el mes, ya que la dedicación a este estudio es, compartida con otras obligaciones externas. Para cada mes, se indica la tarea asignada, la macroetapa en la que se incluye y las fechas estimadas de control.

Mes	Tarea	Macroetapa	Supervisión
Septiembre 2.015	Extracción de requisitos y toma de contacto	1	Raquel García
Octubre 2.015	Planificación del proyecto	1	Raquel García
Noviembre 2.015	Análisis de alternativas RDBMS y herramientas visuales	2	Raquel García
Diciembre 2.015	Análisis Oracle	2	Raquel García
Enero 2.016	Análisis Oracle	2	Raquel García
Febrero 2.016	Elaboración de casos de uso	2	Raquel García
Marzo 2.016	Test en SQL Developer	2	Raquel García
Abril 2.016	Test en Toad	2	Raquel García
Mayo 2.016	Comparación global de	2	Raquel García

	herramientas y redacción de conclusiones		
Junio 2.016	Revisión final del estudio y entrega de la documentación	2	Raquel García
Julio 2.016	Defensa y presentación final	3	Cliente (Ana y tribunal)

Tabla 1: Cronograma inicial de actividades

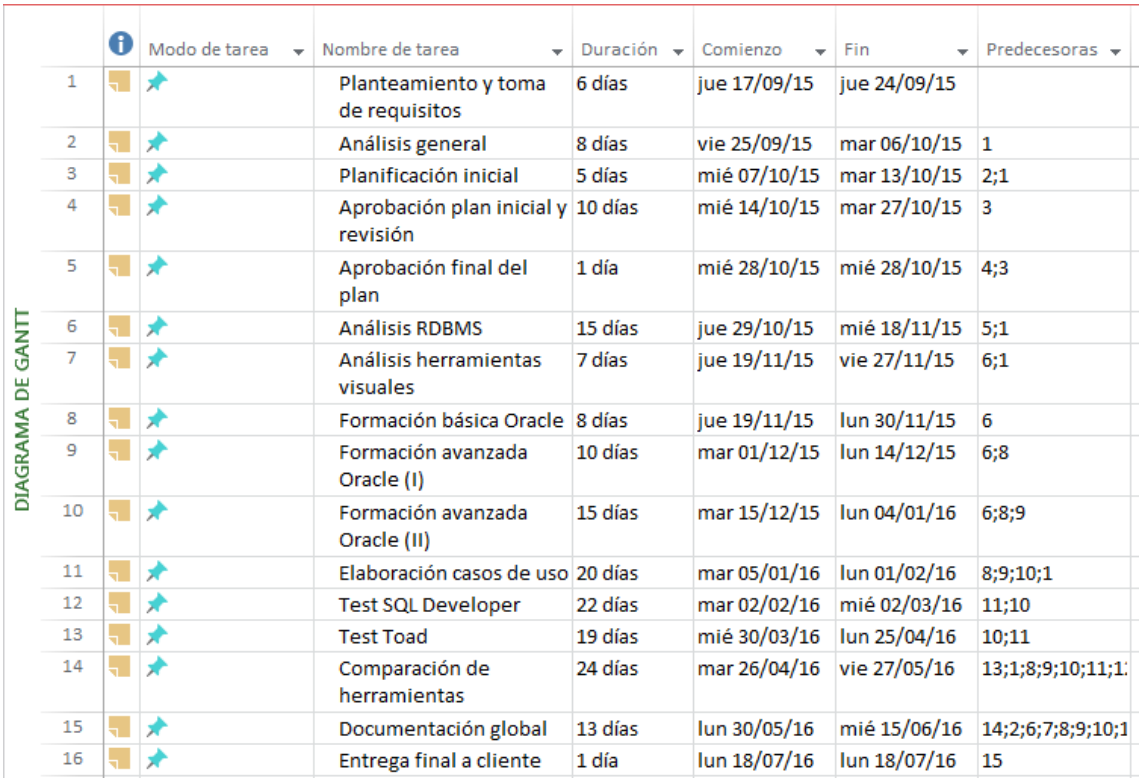


DIAGRAMA DE GANTT

	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1		Planteamiento y toma de requisitos	6 días	jue 17/09/15	jue 24/09/15	
2		Análisis general	8 días	vie 25/09/15	mar 06/10/15	1
3		Planificación inicial	5 días	mié 07/10/15	mar 13/10/15	2;1
4		Aprobación plan inicial y revisión	10 días	mié 14/10/15	mar 27/10/15	3
5		Aprobación final del plan	1 día	mié 28/10/15	mié 28/10/15	4;3
6		Análisis RDBMS	15 días	jue 29/10/15	mié 18/11/15	5;1
7		Análisis herramientas visuales	7 días	jue 19/11/15	vie 27/11/15	6;1
8		Formación básica Oracle	8 días	jue 19/11/15	lun 30/11/15	6
9		Formación avanzada Oracle (I)	10 días	mar 01/12/15	lun 14/12/15	6;8
10		Formación avanzada Oracle (II)	15 días	mar 15/12/15	lun 04/01/16	6;8;9
11		Elaboración casos de uso	20 días	mar 05/01/16	lun 01/02/16	8;9;10;1
12		Test SQL Developer	22 días	mar 02/02/16	mié 02/03/16	11;10
13		Test Toad	19 días	mié 30/03/16	lun 25/04/16	10;11
14		Comparación de herramientas	24 días	mar 26/04/16	vie 27/05/16	13;1;8;9;10;11;1
15		Documentación global	13 días	lun 30/05/16	mié 15/06/16	14;2;6;7;8;9;10;1
16		Entrega final a cliente	1 día	lun 18/07/16	lun 18/07/16	15

Ilustración 1: Gantt inicial (I)

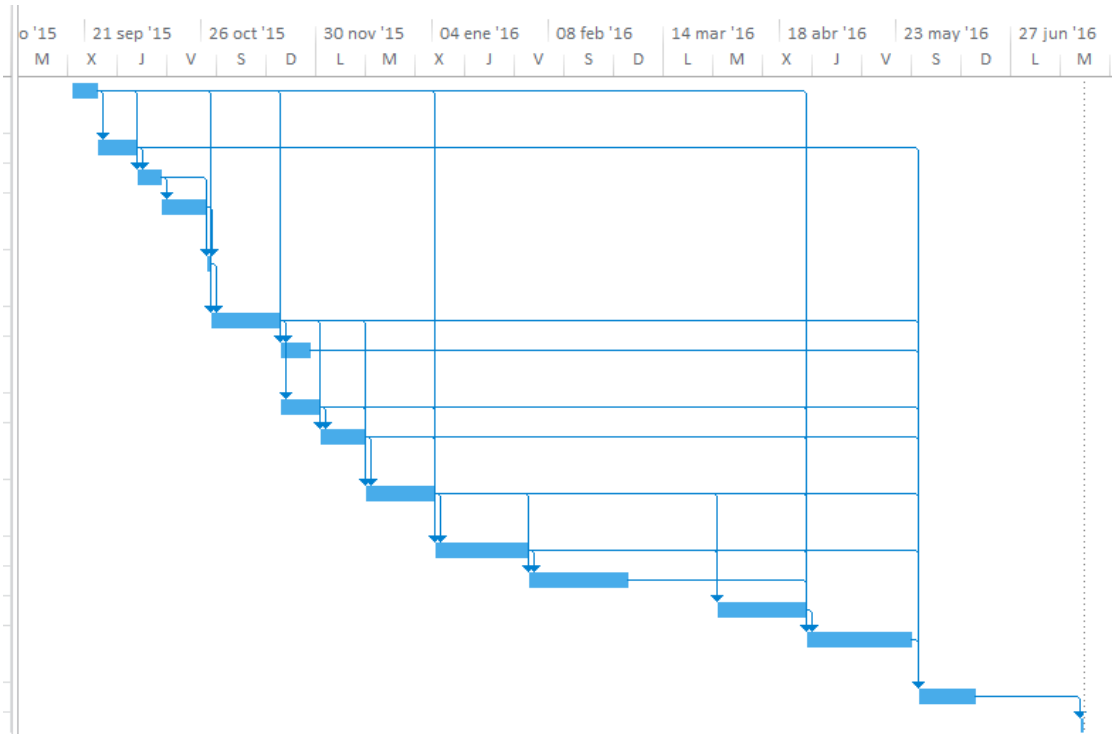


Ilustración 2: Gantt inicial (II)

Todo ello quedaría incluido en un presupuesto total de 19.500 € sujeto a las siguientes condiciones:

- A dicho precio se le añadiría su IVA (actualmente, 21%): 4.095 €. Este valor es aproximado, y podría modificarse en el futuro si el valor del IVA lo hiciera.
- Las funciones analizadas son aquellas disponibles de forma gratuita. En caso de precisar un análisis más exhaustivo que incluya funciones avanzadas, se añadirían los costes de licencias. Como aproximación, estas serían:
    - Toad for Oracle: en concreto se necesitaría el producto "Toad for Oracle Xpert Edition", que incluye las funciones de "Toad for Oracle Professional" y el optimizador SQL. El coste actual de este producto es de 2.772,11 €.
    - Oracle: la versión XE es gratuita y no ofrece la posibilidad de activar el paquete de tuning y diagnósticos, por lo que el sobrecoste de elegir esta opción incluiría la compra de la edición Enterprise (EE) y las licencias de los paquetes. Eso significaría un total de [47.500$ (Oracle EE) + 10.450$ (actualizaciones, licencias y soporte)] + [7.500$ (Diagnostics Pack) + 1.650$ (actualizaciones, licencias y soporte)] + [5.000$ (Tuning Pack) + 1.100$ (actualizaciones, licencias y soporte)] = 73.200$. *Precios actualizados para el periodo 2.015 / 2.016, sujetos a variación.
- Las condiciones de pago son 50% inicial y 50% final, por lo que deberán efectuarse dos pagos, uno al inicio del proyecto y otro al final, de 9.750 €.

2.3 Ejecución Final y Análisis de Costes: GANTT final, Costes de recursos humanos, software y hardware

Los costes finales asociados al proyecto han sido 15.704,45€, asociados al tiempo extra requerido por el estudio. Como desglose de costes se presenta a continuación la siguiente tabla resumen:

Concepto	Unitario	Total absoluto (con amortización)	Total real (del proyecto)
Raquel	20 €/h	320 h	6.400 €
Ordenador	1.250 €	572,90 €	171,88 €
Windows + Licencias Office	280 € + 80 €	61,10 €	18,33 €
Material fungible	250 €	250 €	75 €
Impresora + recambios	150 € + 200 €	227,5 €	68,25 €
Viajes y dietas	0	0	0
Electricidad	500€ / 2 meses	2.750 €	825 €
Agua	100€ / mes	1.100 €	330 €
Alquiler	1.400 € / mes	15.400 €	4.620 €
Internet	70 € / mes	770 €	231 €
Salarios indirectos	1.000 / mes	11.000 €	3.300 €
Total			16.039,46 €

Tabla 2: Desglose de costes

En software: licencias de Microsoft Office para utilizar el MS-Project, el Excel para manejar los datos de los casos de prueba (generar el dataset empleado), y el Word para redactar el proyecto. El restante software es de licencia gratuita o ha sido utilizado en licencia temporal.

En hardware: Ordenador portátil HP Pavilion dv6 Notebook PC de 2009, con procesador Intel® Core ™ i7-2670QM CPU @ 2.20GHz 2.20 GHz, Sistema Operativo de 64bits.

Frente a la cronología inicial de las tareas, se adjunta la cronología real. Han surgido dos periodos de pausa, uno en diciembre y otro entre abril y junio, causados por la necesidad de atender otros proyectos con carácter más urgente. Sin embargo esta demora sólo ha supuesto un retraso en la entrega de 2 meses (un 18% más de lo previsto) y un sobrecoste del 9%.

Mes	Tareas	Macroetapa	Supervisión
Septiembre 2.015	Extracción de requisitos y toma de contacto	1	Raquel García
Octubre 2.015	Planificación del proyecto	1	Raquel García
Noviembre 2.015	Análisis de alternativas RDBMS y herramientas visuales	2	Raquel García
Diciembre 2.015	Pausa del proyecto	No aplica	No aplica
Enero 2.016	Análisis Oracle	2	Raquel García
Febrero 2.016	Análisis Oracle	2	Raquel García
Marzo 2.016	Elaboración de casos de uso	2	Raquel García
Abril 2.016	Pausa del proyecto	No aplica	No aplica
Mayo 2.016			
Junio 2.016			
Julio 2.016	Repaso del avance y Test en SQL Developer	2	Raquel García
Julio 2.016	Test en Toad	2	Raquel García
Agosto 2.016	Comparación global de herramientas y redacción de conclusiones	2	Raquel García
Septiembre 2.016	Revisión final del estudio y entrega de la documentación	2	Raquel García
Octubre 2.016	Defensa y presentación final	3	Cliente (Ana y tribunal)

Tabla 3: Cronograma final de actividades

El Gantt real asociado a este proyecto ha sido:

DIAGRAMA DE GANTT

	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1		Planteamiento y toma de requisitos	6 días	jue 17/09/15	jue 24/09/15	
2		Análisis general	8 días	vie 25/09/15	mar 06/10/15	1
3		Planificación inicial	5 días	mié 07/10/15	mar 13/10/15	2;1
4		Aprobación plan inicial y revisión	10 días	mié 14/10/15	mar 27/10/15	3
5		Aprobación final del plan	1 día	mié 28/10/15	mié 28/10/15	4;3
6		Análisis RDBMS	15 días	jue 29/10/15	mié 18/11/15	5;1
7		Análisis herramientas visuales	7 días	jue 19/11/15	vie 27/11/15	6;1
8		Formación básica Oracle	8 días	jue 19/11/15	lun 30/11/15	6
9		Formación avanzada Oracle (I)	14 días	lun 18/01/16	jue 04/02/16	6;8
10		Formación avanzada Oracle (II)	17 días	vie 05/02/16	lun 29/02/16	6;8;9
11		Elaboración casos de uso	20 días	mar 01/03/16	lun 28/03/16	8;9;10;1
12		Test SQL Developer	22 días	lun 04/07/16	mar 02/08/16	11;10
13		Test Toad	19 días	mié 30/03/16	lun 25/04/16	10;11
14		Comparación de herramientas	24 días	mié 03/08/16	lun 05/09/16	13;1;8;9;10;11;12
15		Documentación global	13 días	mar 06/09/16	jue 22/09/16	14;2;6;7;8;9;10;11
16		Entrega final a cliente	1 día	mar 11/10/16	mar 11/10/16	15

Ilustración 3: Gantt final (I)

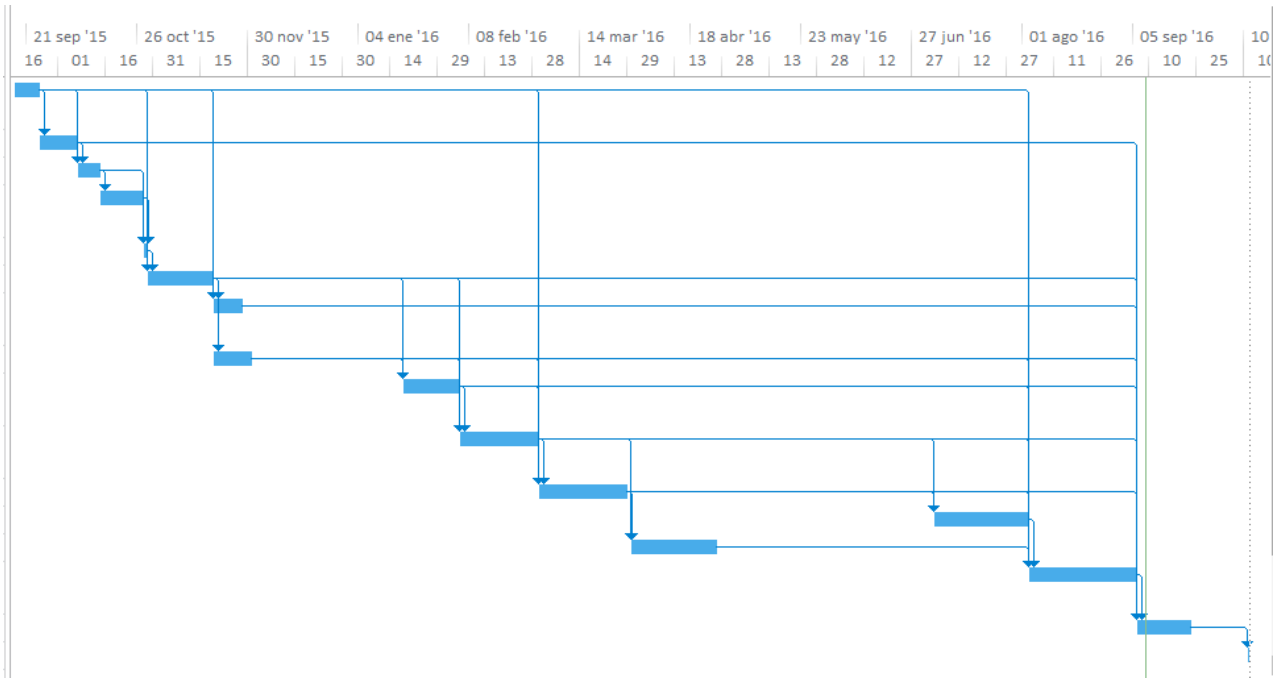


Ilustración 4: Gantt final (II)

3. Estado del Arte

Esta tercera sección está dedicada al análisis previo realizado acerca del entorno en el que se enmarca este proyecto.

Las secciones 3.1 y 3.2 se darán una visión actual del entorno socioeconómico del sector y de las regulaciones existentes, especialmente centradas en el ámbito español y europeo.

En tercer lugar se defenderán el conjunto de alternativas disponibles y finalmente seleccionadas en lo que respecta a bases de datos (relacionales versus no relacionales), sistemas gestores de bases de datos relacionales (RDBMS) y herramientas visuales.

Finalmente se delimitará el alcance y los objetivos del proyecto, sus implicaciones prácticas y el enfoque global adoptado para llevarlo a cabo.

3.1 Entorno socio-económico

La creación, administración y desarrollo de las bases de datos se incluye en el denominado '*Sector IT*' por sus siglas en inglés: *Information Technologies* o Tecnologías de la Información. En este estudio se empleará su término anglosajón, más extendido.

Tradicionalmente se distinguen cuatro generaciones en la evolución de los computadores según fuera su componente clave. Esta industria se ha diversificado cada vez más y ahora cubre aspectos como el hardware, el diseño gráfico, las redes, el software, los sistemas operativos, los lenguajes de programación y las bases de datos. En los últimos años se ha invertido un gran esfuerzo en desarrollar las comunicaciones, debido al poder de la información, equiparable al que anteriormente tenían las materias primas y más tarde las industrias.

En torno a este concepto surgieron las '*Information Technologies*', encargado de recabar, almacenar, tratar, y usar la información mediante la integración de las estructuras informativas y los datos disponibles en el mundo real con los equipos electrónicos (esencialmente ordenadores), con el fin de obtener una utilidad. Esta utilidad puede ser el mero almacenamiento, conservación y divulgación de la información, pero también el procesamiento de datos que permita tomar mejores decisiones cuando son interpretados dentro de un contexto concreto.

Dentro del amplio espectro de actividades, actualmente las que más desarrollo están teniendo son el análisis de datos, la mejora de las comunicaciones y la productividad, y la ciberseguridad.

El sector de la información es el que tiene proyectado un mayor crecimiento en cuanto a generación de riqueza con una tasa del 4,7% anual (US Bureau of Labor Statistics, 2012), en comparación con el 2,3% anual entre 2.000 y 2.010. Sin embargo

también es una de las industrias con el declive del producto más rápido. Según el estudio llevado a cabo por PwC, el 67% de los CEOs participantes tienen planeado contratar más personal este año, intención un 12% superior a la del anterior año y la más alta desde hace seis. La atracción y retención del talento aparecen en segunda posición en la lista de prioridades dirigidas a mejorar los niveles de crecimiento. Dos tercios de los CEOs son conscientes de que cada vez más profesionales prefieren trabajar en empresas cuyos valores compartan y el 75% ya está tomando medidas (PwC, 2.016).

La importancia del sector radica en su capacidad de transformación en todos los niveles. Se ha erigido como motor del cambio y la mejora social, de ahí que la inversión y desarrollo de esta industria haya aumentado su peso dentro del PIB de algunos países. Sin embargo, requiere una inversión inicial en infraestructuras y educación a largo plazo. El 90% de la población de los países más pobres y el 60% de la población mundial no está aún conectada (Cornell University, World Economic Forum, INSEAD, 2.015).

Aunque es un sector extendido por todo el mundo, existen cinco países especialmente significativos: Estados Unidos, Reino Unido, India, China y Alemania. Considerando el resto del globo, existe una clara diferencia entre países desarrollados y en vías de desarrollo en cuanto a la capacidad de maximizar el uso de las tecnologías de la información y comunicación. El mismo informe sitúa entre los diez primeros puestos del ranking a siete países europeos y sólo a uno de los 'tigres asiáticos' (Singapur), junto con Japón y Estados Unidos. Dentro de Europa, los países nórdicos como Finlandia, Suecia y Noruega están a la cabeza. Algunos se han mantenido en sus posiciones (Irlanda, Malta, España y Chipre), mientras que otros la han mejorado: Portugal, Italia y Grecia. El resto de regiones presenta diferencias aún mayores entre los mejores y los peores países, excepto el África subsahariana, que agrupa la mayoría de los países con peores resultados (Cornell University, World Economic Forum, INSEAD, 2.015).

En España la crisis de 2.007 obligó a reducir las inversiones en investigación y en el mantenimiento de las infraestructuras, lo que en conjunto llevó a una disminución del valor añadido del sector IT hasta el 2.013. También el número de ocupados, las ganancias medias por trabajador y mes, y los indicadores de I+D se vieron afectados (INE España). En el caso del número de ocupados en el sector, a pesar de que en el sector en su conjunto se ha reducido, han aumentado los empleados en empresas de programación y consultoría. Por su parte, los salarios han aumentado en todas las ramas, pero en una cantidad insuficiente en comparación con el encarecimiento del nivel de vida. La evolución del número de empresas es un caso especial, y en la actualidad las empresas dedicadas al I+D además de sufrir recortes en gastos en inversión han experimentado un descenso del número de empresas (INE España).

Sin embargo, la reactivación de la economía en los países occidentales está cambiando la situación. Esto unido a la resistencia de las empresas del sector durante este período ofrece unas perspectivas de crecimiento y generación de empleo

inimaginables hace cinco años. Una señal de este cambio es la creciente inversión que se está realizando en este sector en España: "Respecto a idéntico periodo de 2014, los fondos de capital riesgo han elevado el valor de sus adquisiciones y rondas de financiación en este sector un 222%" (Expansión, 2.015). Internet y el desarrollo del software son las dos áreas tecnológicas que más inversión reciben.

3.2 Marco Regulador

En la actualidad la regulación de las tecnologías de la información, en la que se incluiría todo el espectro de actividades relacionadas con las bases de datos, gira en torno a los conceptos de software propietario y software libre.

Richard Stallman definió en 2.002 el software libre como aquel que proporciona cuatro libertades básicas: 1) Libertad para ejecutar el código, independientemente del lugar, propósito y duración; 2) Libertad para estudiarlo y adaptarlo a las propias necesidades; 3) Libertad para redistribuirlo y formar redes de colaboración entre los usuarios; y 4) Libertad para mejorar el código y publicarlo.

El software libre aporta un modelo de costes más competitivo que el propietario al nacer de la colaboración entre desarrolladores y usuarios, facilita el desarrollo de nuevos programas que reutilicen parte del código, la distribución y expansión vía Internet y contenidos generados por los usuarios reduce los costes de publicidad y distribución tradicionales, y lo niveles de calidad esperados son superiores gracias a la mejora continua generada por una mezcla de colaboración y competición entre los involucrados.

En el otro extremo, el software propietario tiene mayor presencia en entornos corporativos debido a la falta de confianza en el libre, ya que la información que se maneja en ellas es esencialmente confidencial.

Existen cinco factores clave en la diferenciación entre ambos tipos de software:

1) Costes: los propietarios suponen un coste monetario en cuanto al pago de licencias y adquisición de servicios y equipamiento adicionales superior al libre. Sin embargo, este segundo requiere una inversión superior en términos de aprendizaje, instalación y mantenimiento (al no existir oficialmente un único responsable de desarrollo, no hay garantías de que el software vaya a ser mantenido en el futuro).

2) Adaptabilidad y capacidad de desarrollo: muchos software propietarios funcionan como una caja negra, lo que imposibilita en gran medida al usuario una adaptación ad-hoc. Incluso en el caso de que permitan aplicar modificaciones, estas suelen ser limitadas. Los software libres se situarían en el otro extremo, por lo que el ajuste entre funcionalidad aportada por el software y las necesidades del usuario es mayor.

3) Dependencia del proveedor: en la modalidad libre el equipo de desarrollo no está concentrado en un único grupo, sino que al ser producto de la colaboración la relación de dependencia es inferior a la existente en uno propietario. En estos últimos, los problemas que afectaran al equipo de desarrollo tendrían serias consecuencias en los clientes, lo que haría peligrar la continuidad del producto, sus mejoras y mantenimiento. Por otro lado, las actualizaciones para hacer frente a cambios legislativos, mejorar la seguridad o resolver vulnerabilidades, y la disponibilidad de un soporte especializado están más garantizadas en los entornos propietarios.

4) Modularidad: el grado de modularidad suele ser mayor en código abierto, debido al gran número de colaboradores que intervienen (MacCormack et al. 2006). La ventaja de este enfoque es la facilidad para desarrollar nuevas funciones, sin necesidad de contar con todo el código del software, lo que reduce la complejidad. Además, desde el punto de vista de la seguridad, la modularidad permite encapsular vulnerabilidades y sus respectivas correcciones.

5) Calidad: Boulanger (2.005) defiende el código libre en cuanto a calidad y fiabilidad una vez se ha alcanzado una cantidad suficiente de seguidores que se involucren en la mejora del producto. Sin embargo esta idea se basa en la confianza que depositen los usuarios en la comunidad y en su deseo de aprovechar las potenciales sinergias, características que se pierden cuando el valor de la información y nivel de confidencialidad son elevados.

La propiedad intelectual es un concepto muy ligado al software propietario, y es lo que motiva la aparición de sus licencias de uso. En un software libre, aunque incluya información sobre el desarrollador, no se impide que pueda reutilizarse parte de dicho código para generar un nuevo producto (es una de las cuatro libertades características de este software). Sin embargo, en uno propietario, los derechos de uso están restringidos por licencia. En algunos casos se opta por una mezcla, ya sea ofreciendo herramientas con funciones gratuitas limitadas (SQL Developer) o con periodos de prueba (TOAD).

En España, la Oficina Española de Patentes y Marcas (en adelante, OEPM) denomina a las 'patentes de software' como 'invenciones implementadas en ordenador': "Aquella que implica el uso de un ordenador, una red informática u otro aparato programable en el que una o más de sus funciones se llevan a cabo total o parcialmente gracias a un programa de ordenador" (OEPM). Respecto a la legislación vigente, los derechos de autor sobre los programas de ordenador viene regulado por los preceptos del Real Decreto legislativo 1/1.996, de 12 de abril, en el que se aprobó el texto refundido de la Ley de Propiedad Intelectual. En cambio, el Convenio de la Patente Europea (CPE) no contempla un programa de ordenador como invención patentable.

Actualmente, la Oficina Europea de Patentes (OEP) cuenta con una de las mayores bases de datos de patentes del mundo. Además de servir como protección del desarrollo y garantía de la recuperación de la inversión, las patentes sientan las bases para nuevos desarrollos y contribuyen a la base de conocimiento de una sociedad. Con

este objetivo, el Convenio de Patentes Europeo (CPE) otorga a todo individuo el derecho a solicitar una patente, independientemente de si se trata un particular, una PYME (Pequeñas y Medianas Empresas) o una gran empresa, aunque como se indicó anteriormente no se puede solicitar sobre cualquier materia.

A esto se le añade el hecho de que las leyes de propiedad intelectual no tienen un alcance homogéneo, por lo que el lugar en el que se patente un determinado producto es decisivo en cuanto a la vigencia de la patente, requisitos para obtenerla y acciones punitivas en caso de ser violada por otro agente. Esta situación se agrava en el sector de la tecnología, cuya rápida evolución dificulta crear leyes que abarquen todos los aspectos, por lo que hoy en día existen grandes vacíos legales. Por ejemplo, en 2.005 el Parlamento Europeo rechazó incluir el software como invención que se pueda patentar, alegando que "las leyes existentes ya protegían adecuadamente las invenciones de software y que las patentes aumentarían el coste de innovar" (Bolaños, 2.015).

El comportamiento del software como caja negra se repite en el caso de las patentes. No existe disposición legal alguna en el CPE que obligue a incluir el código fuente en la patente, sino que se considera suficiente una descripción de la invención implementada en ordenador. Este mismo convenio establece que sólo se concederán patentes a invenciones que cumplan tres requisitos: 1) ser nuevas, 2) suponer una actividad inventiva y 3) ser susceptibles de aplicación industrial (Artículos 54, 56 y 57 del CPE).

Esta situación de incertidumbre legal en la protección del software obliga a las empresas del sector a defenderse mediante dos vías principales: 1) absorbiendo a la competencia adquiriendo nuevos productos que detecten como potenciales sustitutivos, y 2) permitiendo el acceso a sus productos mediante el pago de licencias. Un ejemplo de estos comportamientos es Oracle, que compró en 2.010 Sun Microsystems (creadora del lenguaje JAVA) y en estos momentos es una de las mayores empresas dedicadas al desarrollo de software corporativo.

3.3 Sistemas de almacenamiento de datos

3.1.1 Bases de datos relacionales y bases de datos no relacionales

Actualmente las bases de datos pueden dividirse en dos grandes grupos: relacionales y no relacionales. Las primeras trabajan con datos estructurados a los que se accede mediante consultas también estructuradas (de ahí su nombre alternativo, bases de datos SQL). Las segundas manejan datos semiestructurados, lo que aporta un dinamismo que las anteriores no pueden alcanzar y que se ajusta en mayor medida a la realidad, donde la velocidad a la que se generan nuevos datos crece exponencialmente.

Aunque el término 'NoSQL' pueda parecer novedoso actualmente, este término ya fue empleado en primer lugar por Carlo Strozzi en 1.998 en una base de datos que estaba desarrollando y que no se ajustaba al estándar SQL porque se basaba en archivos en vez de tablas. Hasta 2.009 no se retomó, cuando Eric Evans lo empleó al definir las bases de datos no relacionales, cada vez más presentes, cuyos datos se almacenaban de forma distribuida y no trataban de asegurar las propiedades ACID (atomicidad, consistencia, independencia y durabilidad).

Si las bases de datos relacionales deben cumplir las propiedades ACID para ser consideradas como tal, las no relacionales se basan en las propiedades BASE definidas por Eric Brewer. Estas propiedades parten del teorema CAP, que establece que un sistema distribuido no puede garantizar el cumplimiento simultáneo de la consistencia, la disponibilidad y la tolerancia a particiones; sino que será necesario elegir dos de las tres. El sistema BASE renuncia a la consistencia en beneficio de la disponibilidad y la tolerancia a particiones.

Las bases de datos NoSQL aportan un mecanismo de almacenamiento y extracción de datos que usan modelos de consistencia más ligeros que los tradicionales de las relacionales. Esto significa más simplicidad en el diseño, facilidad para la escalabilidad e incluso mayor control sobre la disponibilidad. Todo ello las convierte en unas bases de datos óptimas para la industria del big data y las aplicaciones web en tiempo real, donde los objetivos de rendimiento están más relacionados con latencia y flujo de información.

Según el Dr. S. George de la Universidad de Nueva Zelanda, se pueden distinguir cuatro aproximaciones distintas para clasificar internamente las bases de datos NoSQL:

- Por columna, como Hbase, BigTable, HyperTable y Accumulo. En lugar de almacenar los datos en filas, estas bases de datos utilizan las tablas de datos como secciones de columnas de datos, es decir, las claves apuntan a múltiples columnas distribuidas entre múltiples máquinas y que se agrupan por familias de columnas. Esto permite alcanzar mayores niveles de rendimiento y escalabilidad.

- Por documentos, como MongoDB y Couchbase. La información se almacena en documentos encapsulada y codificada usando estándares como XML, YAML, JSON, BSON o PDF. Son la evolución de la implementación por columnas, y soportan de manera más eficiente las consultas.

- Por key-valor (valor clave), como Apache Cassandra, Dynamo, Riak, Redis, Cache, BerkeleyDB y Project Voldemort. Almacenan los datos sin requerir esquema alguno, sólo se necesita una clave indexada con un valor asociado que conduce a los datos pedidos. Dentro de esta categoría se incluyen otras subcategorías en función del modelo interno que utilicen, como jerárquico, de caché en RAM, disco rotatorio, etc. Aunque es la implementación más

sencilla, también es muy ineficiente, especialmente cuando sólo se busca cambiar parte de los datos a los que se accede.

- Por gráficos, como Neo4J, Allegro y Virtuoso, que almacenan datos interconectados por un número indeterminado de relaciones. Se utilizan para representar por ejemplo relaciones sociales, enlaces de transporte público, y topologías de red.

En términos de rendimiento, escalabilidad, flexibilidad, complejidad y funcionalidad aportada por cada una de las subcategorías de las bases de datos no relacionales y las relacionales, se muestra a continuación una comparativa general de estos enfoques: [*Fuente: International Journal of Enterprise Computing and Business Systems; ISSN (Online): 2230-8849; Volume 2 Issue 2 July 2013; Dr. S. George - University of New Zealand*]

Data Model	*Performance*	*Scalability*	*Flexibility*	*Complexity*	*Functionality*
KEY-VALUE STORES	High	High	High	None	Variable (none)
COLUMN STORE	High	High	Moderate	Low	Minimal
DOCUMENT STORE	High	Variable (high)	High	Low	Variable (low)
GRAPH DATABASE	Variable	Variable	High	High	Graph Theory
RELATIONAL DB	Variable	Variable	Low	Moderate	Relational Algebra

Tabla 4: Comparativa de tipos de bases de datos relacionales y no relacionales. Fuente: International Journal of Enterprise Computing and Business Systems

Sin embargo, este estudio se ha centrado en las primeras por los siguientes motivos:

- El modelo relacional es fácilmente comprensible tanto en la teoría como en la práctica, y por tanto su utilidad a nivel académico es mayor. Permite establecer una base sólida de conocimientos sobre las bases de datos antes de extenderlo a otras aproximaciones más novedosas.

- El lenguaje SQL utilizado en sus consultas está basado en el Álgebra Relacional, dotándolas de un marco matemático.

- Cumplen las propiedades ACID*: atomicidad, consistencia, aislamiento y durabilidad, lo que las hace especialmente recomendables para bases de datos dedicadas a transacciones de información sensible (por ejemplo, información personal como en las bases de datos corporativas sobre los empleados).

- La mayoría de las bases de datos empresariales continúan siendo relacionales, ya que los datos que gestionan mediante la base de datos son estructurados y las

aplicaciones que los emplean son centralizadas. Además permiten ser más eficientes.

El entorno en el que se desarrolla este proyecto no se ve afectado por las limitaciones de este tipo de bases de datos, como puede ser la imposibilidad de realizar búsquedas en datos no estructurados o la recepción de datos desde diversos puntos debido a la descentralización de las aplicaciones que compartan la base de datos, limitaciones esencialmente relacionadas con Internet.

3.1.2 Sistemas gestores de bases de datos relacionales

Una vez centrado el proyecto en las bases de datos relacionales, el siguiente paso es seleccionar el RDBMS o Sistema gestor de base de datos relacionales (SGBDR). Este será el punto de contacto ente el usuario (en este caso el administrador del sistema) y la base de datos en sí. Aunque no todos utilizan el mismo lenguaje, el SQL tiene una gran presencia entre ellos al haber sido adoptado como estándar.

A continuación se dará una breve descripción de los sistemas gestores más conocidos, y que han formado parte del grupo de alternativas consideradas en el planteamiento de este estudio. Se revisarán Oracle, DB2, SQL Server, MariaDB, MySQL, PostgreSQL, Apache Derby y Amazon Aurora.

Oracle DB XE 11g R2: Es una base de datos gratuita desarrollada por Oracle y especialmente recomendada para principiantes y estudiantes que acaben de empezar a estudiar las bases de datos, publicada en junio de 2.014. Como su propio nombre indica, "Express Edition", es una base de datos más ligera que otras versiones, y que por tanto tiene las siguientes limitaciones de hardware: 1) sólo emplea un núcleo de la CPU, independientemente de que tenga más (por ejemplo, que sea dual-core), 2) sólo permite utilizar 1GB de memoria RAM incluso aunque haya más disponible, 3) sólo es posible ejecutar una instancia en cada ordenador, 4) el máximo de datos de usuario que puede almacenar es 11GB, y 5) no soporta HTTPS. Todas estas restricciones se solventarían utilizando versiones Standard o Enterprise o añadiendo un web listener alternativo como Apache en el último punto, pero en el caso de este estudio no es necesario. A cambio ofrece una herramienta gratuita, respaldada por una empresa reconocida a nivel mundial, y que cuenta con una vía de soporte para los usuarios que lo requieran a través de sus foros.

Existen otros productos de Oracle como Oracle Standard Edition (SE) o Enterprise Edition (EE), que no tendrían las restricciones hardware que tiene la versión XE. Sin embargo, su instalación es más costosa en términos de conocimientos técnicos necesarios y recursos, y los escenarios para los cuales están diseñadas no son el de este proyecto.

Todas las versiones de Oracle DB tienen un requisito común: el paquete Oracle Diagnostics Pack es indispensable y bajo licencia de pago si se quieren utilizar las

funcionalidades del Oracle Tuning Pack. Algunas de estas funcionalidades son SQL Access Advisor, SQL Tuning Advisor y Automatic SQL Tuning, todas ellas relacionadas con la optimización de consultas. Información adicional sobre estas licencias se puede encontrar en la web oficial de Oracle.

DB2: RDBMS de IBM en 1.983 creada para ser ejecutada en su plataforma MVS (Multiple Virtual Storage). Entonces fue considerada una evolución del modelo jerárquico de las bases de datos al recién aparecido modelo relacional. Aunque comenzó sólo en plataformas de IBM, posteriormente amplió sus compatibilidades con los principales sistemas operativos como UNIX, Windows y Linux y en la actualidad ha aumentado este abanico de plataformas, convirtiéndola en una de sus principales ventajas competitivas. Su gestión se basa tanto en línea de comandos como en interfaz gráfica, y es una de las más extendidas por su entorno de seguridad, aspecto crítico en el mundo corporativo.

Como inventor del lenguaje SQL, IBM cuenta con un dialecto de SQL más potente que el de su competidor Microsoft y al contrario que MS SQL, DB2 soporta métodos Java y arrays, y funciones definidas por múltiples usuarios. Incluso acepta partes de código escritas en otros lenguajes como Java o COBOL. Tiene además integrado soporte a otras competencias NoSQL, como son XML, el almacenamiento de gráficos y la notación JSON (Java Script Object Notation).

Está diseñada tanto para operaciones transaccionales como analíticas, mientras mantiene una disponibilidad continua de los flujos de trabajo.

Sus detractores destacan la incapacidad para ejecutar nodos secundarios sin incurrir en nuevos costes e incluso herramientas de terceros, y no está preparada para gestionar páginas o servicios web. Además, el mantenimiento de esta base de datos es complejo si no se ha contratado el servicio oficial de soporte de IBM y los costes de licencias elevados.

SQL Server: es la propuesta relacional de Microsoft. Su política de código privado y pago obligatorio para todos sus componentes (hardware, licencias, soporte, etc.) daña la opinión entre los usuarios sobre ella. A su favor tiene el estar respaldada por una de las mayores empresas del sector tecnológico a un coste inferior de Oracle, lo que es garantía de un estándar de calidad y capacidad de integración adecuada con otros productos Microsoft. Se recomienda para empresas PYMES, cuyos presupuestos son mayores que los de un particular pero están muy por debajo de los de las grandes empresas. Salvando el obstáculo del coste, las opiniones sobre su rendimiento están divididas en la comunidad, y dependen del fin que tenga la base de datos. Sólo es apto para operar en plataforma Windows, no acepta particiones y la visualización del contenido es mejorable.

MySQL: También propiedad de Oracle (y antes de Sun Microsystems), es una base de datos relacional open-source distribuida bajo una licencia 'copyleft' (más restrictiva que la BSD de Postgre SQL). El ámbito de aplicación de esta base de datos es

el de aplicaciones web con operaciones esencialmente de lectura, con consultas más sencillas. Actualmente es una de las bases de datos de código libre más extendidas a nivel empresarial, aunque su desarrollo es privado. En cuanto a su documentación, existe descontento en cuanto al nivel de detalle en sus 'release notes', que en numerosas ocasiones están además incompletas. Incorpora plugins externos sólo disponibles a través de terceros, pero no están integrados en el sistema como hace MariaDB. Su mayor factor en contra es la opinión pública sobre Oracle y sus intenciones de mantener este producto. Un ejemplo fue Google, que en 2.013 decidió migrar sus sistemas internos a MariaDB en lugar de MySQL.

MariaDB: Surgió en 2.009 a raíz de MySQL, en respuesta de los creadores de este al ver cómo Oracle después de adquirirlo no apoyó su desarrollo todo lo esperado. Sus versiones son homónimas a las de MySQL al tiempo que mantiene su compatibilidad con ella. Tanto su distribución como su desarrollo son abiertos, por lo que sus mejoras aparecen a un ritmo superior. Lo mismo sucede con su documentación. Una de sus últimas incorporaciones es incluir en la misma interfaz control tanto de software SQL como no-SQL, para más facilidad de uso. La velocidad de procesamiento y el optimizador de consultas son levemente mejores que MySQL, aunque esta diferencia aparece magnificada en algunos estudios de benchmark.

PostgreSQL: Es una base de datos relacional, de código abierto, y disponible mediante licencia BSD (Berkely Software Distribution). Esta licencia no se refiere al pago por el uso del software, sino que está pensada para proteger la autoría original, de forma que cualquier ampliación hecha sobre este código deba reconocer el origen, por lo que el código es de libre acceso. Dentro del subgrupo de RDBMS de código abierto, esta es considerada la más potente. McAfee, Trend Micro - Tipping Point, Skype o la Autoridad de Aviación Federal estadounidense son sólo algunas de la empresas cuyos productos emplean esta base de datos.

Se basa en un modelo cliente - servidor con multiprocesos para aportar estabilidad. Al igual que DB2, es compatible con los principales sistemas operativos: Linux, UNIX y Windows. Destacan a su favor las siguientes características: base de datos ACID, control de concurrencia multiversión (MVCC), soporta objetos pesados de 64 bits, incluye diversos métodos de autenticación, permite hacer copias de seguridad en caliente, es capaz de ejecutar funciones y procedimientos en numerosos lenguajes (PL/pgSQL, PL/Perl, PL/Python, PL/Tcl), atomicidad a nivel de sub-transacciones, y abarca una gran variedad de posibilidades SQL, entre otras. Todo ello busca centrarse en desarrollar la fiabilidad, la integridad de los datos y el desarrollador. Sus últimas aportaciones se han orientado a mejorar la velocidad de proceso y otras características específicas del mundo empresarial como el manejo de grandes volúmenes de datos.

En comparación con otras bases de datos como MySQL, se mencionan la lentitud de las operaciones 'insert' y 'update', un consumo mayor de recursos y algunas sintaxis menos intuitivas.

Por último, se añaden dos sistemas adicionales que completan esta revisión de las alternativas actuales disponibles:

Apache Derby: Es un subproyecto de ApacheDB, de código abierto e implementado completamente en Java. Proporciona un driver JDBC que permite incluir Derby en cualquier solución basada también en Java. Al igual que Maria DB se centra en la facilidad de instalación, desarrollo y uso. A pesar de ser ligero y soportar la arquitectura cliente - servidor, no existen muchos estándares en la industria orientados a objetos, por lo que su popularidad es menor que el resto de alternativas consideradas. Se ha incluido en el estudio con el fin de dar una visión más completa de la diversidad existente en el sector de las bases de datos relacionales.

Amazon Aurora: Motor de base de datos relacional compatible con MySQL que pretende ofrecer la velocidad y disponibilidad de bases de datos comerciales y la sencillez y el coste de bases de datos de código abierto. Amazon RDS (Relational Database Service), sobre lo que trabaja Aurora, es capaz de combinar varios motores de bases de datos (MySQL, Oracle, Microsoft SQL Server y PostgreSQL). Gestiona tareas rutinarias de la base de datos, como el aprovisionamiento, las revisiones, las copias de seguridad, la detección de errores y la reparación. Ser compatible con cualquier operación de MySQL hace que los cambios de código sean innecesarios. Este producto forma parte de los servicios web de Amazon (‘AWS’, ‘Amazon Web Services’), por lo que no es posible trabajar con una copia en local, es necesario confiar en sus estructuras.

Como resumen de las características de los anteriores sistemas se presenta la siguiente tabla:

Característica	OracleXE 11g r2	DB2	SQL Server	MariaDB	MySQL	PostgreSQL
Ver plan de ejecución de una consulta	Sí	Sí	Sí	Sí	Sí	Sí
Propietario vs Open-Source	Mezcla (hay funcionalidades de pago)		Propietario	Open-Source	Open-Source	Open-Source, Licencia BSD
Compatibilidad	Windows, Linux	Windows, Linux, Unix	Sólo con productos Microsoft	MySQL, software SQL y no-SQL, numerosos motores de almacenamiento	Linux, Windows, Apple OS, IBM AIX, Solaris, Debian, Red Hat	Windows, Linux, Unix
Soporte	Necesario a nivel profesional	Necesario a nivel profesional	Disponible	Colaborativo	Colaborativo	Disponible
Facilidad de aprendizaje (recursos y comunidad)	Amplia documentación, foros, escasez de ejemplos	No	Sí, sobre todo si el usuario no es experto	Sí	No	Sí
Lenguaje(s)	SQL, PL/SQL, SQL*Plus	SQL	TSQL	SQL	SQL	SQL, PL/pgSQL, PL/Perl, PL/Python
Tipos de datos	Numeric, character, date/time	Numeric, character, date/time, LOBs, XML, user-defined; JSON	Números, textos, fechas, xml, binarios, geográficos, geométricos, jerárquicos	Tinyint, Boolean, int, smallint, mediumint, bigint, decimal, float, double, bt, string, char, varchar, binary, blob, text, enum, date, time…	Inteet, bit, tinyint, smallint, bigint, decimal, float, real, datetime, money, image,…	Numéricos, monetarios, de carácter, binaries, fechas y tiempo, Boolean, enumerados, geométricos, xml, json, arrays, personalizados

Tabla 5: Comparativa de RDBMS

Los laboratorios Gartner publican anualmente sus conocidos cuadrantes que sitúan empresas y bases de datos en función de la capacidad de ejecución y completitud de visión. El resultado de 2.015 fue el siguiente:

Ilustración 5: Informe Gartner 2.015

En función de las opciones consideradas y el estudio Gartner, este proyecto se centra en Oracle, concretamente en la versión XE 11g reléase 2, ya que además al ser una de los sistemas más populares existe una gran variedad de herramientas gráficas compatibles con él. Las características concretas de la versión concreta usada para este estudio pueden consultarse en la vista dinámica v$version:

```
SQL> select * from v$version;

BANNER
--------------------------------------------------------------------------------
Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production
PL/SQL Release 11.2.0.2.0 - Production
CORE    11.2.0.2.0      Production
TNS for 64-bit Windows: Version 11.2.0.2.0 - Production
NLSRTL Version 11.2.0.2.0 - Production
```

Ilustración 6: Versión Oracle

3.1.3 Constructores visuales de consultas y administradores de bases de datos gráficos

Este apartado está dedicado a describir las principales herramientas gráficas candidatas a ser usadas como complementarias del RDBMS Oracle XE 11g. El objetivo es seleccionar dos de las cuatro que más ayuda presten a la mejora de las consultas, que posteriormente serán utilizadas en los tres casos de uso establecidos en la sección 4.4. Los resultados específicos de cada una de las dos herramientas serán presentados en las secciones 4.5 y 4.6, para ser comparados y valorados finalmente en la sección 5 del proyecto.

Aunque existe una gran variedad de herramientas gráficas disponibles compatibles con Oracle finalmente han sido objeto de estudio dbForge, Navicat, SQL Developer, TOAD for Oracle y MyOra.

dbForge: Desarrollado por Devart, esta herramienta de desarrollo de bases de datos gráfica ayuda a mejorar la rapidez de codificación PL/SQL y el manejo de datos tanto externos como internos de la base de datos. Su interfaz gráfica es consistente con Microsoft Visual Studio, por lo que sólo es compatible con el sistema operativo Windows. Respecto a la optimización de consultas, la versión 3.8 (Febrero de 2.016) emplea la funcionalidad Oracle SQL Profiler para analizar y monitorizar las consultas. Permite obtener el plan de ejecución y las estadísticas de sesión de la consulta, mostrándolas gráficamente y de forma amigable. Para dos ejecuciones independientes, permite comparar tanto las estadísticas de sesión como las propiedades de monitorización. Este último aspecto es útil para comprobar si otros cambios han sido aplicados y han provocado una mejora o desmejora sensible en el rendimiento.

Entre sus principales ventajas está la usabilidad de la interfaz. Los esquemas son presentados en forma de tabla, lo que permite situar mejor los datos correspondientes a cada operación, e incluyen una numeración en las operaciones relativa, que guía al usuario en la interpretación del plan de ejecución (la última operación, 'SELECT' en el caso de las consultas aparecerá como 'step 12 of 12', mientras que las anteriores operaciones que hayan tenido lugar serán 'step X of 12', siendo X < 12). A esto se le añade una ventana inferior que explica qué operación ha tenido lugar y su interpretación. En las comparaciones, no sólo es capaz de guardar los datos de cada ejecución, sino el texto de la consulta, lo que evita al usuario crear tantas versiones de la consulta como pruebas quiera comparar.

Sin embargo, sólo permite comparar simultáneamente dos ejecuciones y esta funcionalidad no viene incluida en su versión express. Aunque contiene documentación oficial actualizada sobre cada uno de sus productos, el nivel de detalle es limitado, y está pensado para usuarios principiantes, por lo que sólo muestran cómo navegar por la interfaz, pero no ahonda en la interpretación de los resultados ni ofrece otros recursos recomendables.

Navicat: Es la herramienta desarrollada por PremiumSoft CyberTech, y al igual que la mayoría de las herramientas de base de datos su mayor ventaja de uso es la

aportación de una interfaz de usuario amigable que ayude al usuario a administrar la base de datos. Puede conectar con cualquier base de datos Oracle cuya versión sea al menos 8i. Actualmente su versión Premium ha sido nombrada la mejor solución de administración de base de datos por los lectores en 2.015, lo que demuestra el potencial de la herramienta a pesar de su apariencia simple y engañosa.

Esta plataforma cuenta con 10 años de historia, más de 3 millones de descargas y un nutrido grupo de clientes de más de 150.000 personas, lo que la sitúa dentro de las herramientas más consolidadas. Este apoyo de la comunidad tecnológica es apreciable también por los clientes que han trabajado con ello, entre los que se encuentran Apple, Google, HP, PWC, IBM, Samsung, Porsche, ebay y Sony.

Si algo define a Navicat es la extensa familia de productos preparados para complementar a las principales bases de datos relacionales. Entre ellos están. Navicat Premium, for MySQL, for MariaDB, for PostgreSQL, for SQL Server, for Oracle, for SQLite y Navicat Data Modeler. En concreto, su versión Premium ofrece la posibilidad de acceder simultáneamente a seis bases de datos como las mencionadas en sus productos.

En versiones más recientes de los sistemas operativos (al menos 8.1 para Windows y Linux, y 11.0 o superior para Mac) se ha solucionado una de las grandes limitaciones de Navicat: mientras que en versiones anteriores a pesar de soportar múltiples sentencias SQL sólo mostraban los resultados de la última consulta en caso de haber más de una, ahora se han incluido los multiresultados, por lo que en una sola ejecución es posible obtener los resultados de más de una consulta.

Continuando con las ventajas relacionadas con las consultas, caben destacar además las siguientes:

- Bajo determinadas condiciones, es posible actualizar los datos de la base de datos a partir de los resultados obtenidos de una consulta. Sin embargo esta funcionalidad sólo está disponible en algunos de sus productos, y Navicat for Oracle no es uno de ellos.

- Permite parar la ejecución de la consulta a pesar de no haber finalizado. Al contrario que la anterior funcionalidad, esta sí está disponible en todos los productos Navicat.

- Los resultados pueden ser exportados en forma de tabla, lo que es habitual en otras herramientas, o en un archivo con una extensión determinada. Algunos de los archivos disponibles son .txt, .xls, .doc, .html, .dif, .sql, .xml y .rtf.

Sin embargo, puede encontrar problemas a la hora de devolver grandes cantidades de datos como resultado, para lo cual recomiendan oficialmente introducir límites en las consultas mediante la cláusula ‘LIMIT’. Además, las tareas de backup y restore no incluyen las consultas, y la solución más próxima consiste en utilizar los archivos sql.

En conjunto, según la opinión de expertos como Salvador Castro, aunque Navicat resulte una herramienta competitiva en algunos entornos debido a su licencia más asequible, no llega a alcanzar el nivel ofrecido por Toad, por lo que deberán conocerse los requisitos del sistema sobre el que se quiere aplicar la herramienta para elegir la opción más adecuada.

SQL Developer: Es el entorno de desarrollo integrado (IDE*) creado por Oracle para su producto Oracle Database con un objetivo principal: "ayudar al usuario final a ahorrar tiempo y maximizar los beneficios de haber realizado una inversión en productos Oracle". Es una de las herramientas más utilizada, especialmente por su licencia gratuita y por ser la herramienta ofrecida por Oracle para gestionar sus propias bases de datos. Fue también finalista en los premios a mejor solución para la administración de bases de datos en 2.015.

Actualmente está preparado para soportar las versiones 10g, 11g y 12c de Oracle Database y ser ejecutado en cualquier sistema operativo que sea compatible con Java. Contiene una variedad de funcionalidades según el tipo de usuario que pueda necesitarla (no es exclusiva de los desarrolladores):

- Para los desarrolladores, se pone a su disposición editores de código que trabajen con SQL, PL/SQL, Stored Java Procedures (procedimientos de java), y XML; una ejecución de consultas a partir de la cual es posible obtener el plan de ejecución, la posibilidad de ejecutar, 'debuggear' y probar código, así como exportar datos en múltiples formatos.

- Para los administradores de bases de datos o DBA, desde la versión 3.0 se incluyeron nuevas interfaces dedicadas a tareas críticas como 'Oracle Auditing', gestión de usuarios y roles, gestión de almacenamiento (que incluso permite añadir espacio al tablespace) y la gestión de recursos.

- Para los arquitectos de aplicaciones y modeladores de datos ('*application architect & data modeler*') existe una solución incluida en SQL Developer, que también es accesible independientemente, llamada SQL Developer Data Modeler o SDDM. Esta incluye los modelados lógico, relacional, físico y dimensional, diagramas de flujo de datos, la importación de datos desde fuentes dispares como los diccionarios, scripts DDL y repositorios externos, el control de versiones (Git) y una potente herramienta de búsqueda y generación de informes.

- Para los desarrolladores y administradores de aplicaciones web, SQL Developer también permite administrar los servicios Rest Data de Oracle, crear y modificar los servicios RESTful. SQL Developer es capaz de integrarse con otra herramienta de desarrollo, Oracle APEX*, para navegar, exportar e importar, eliminar o desplegar aplicaciones, y hasta crear sobre todo ello informes personalizados.

- Por último, en caso de que se desee migrar una base de datos, incluyendo aquellas no Oracle, SQL Developer se ha convertido en la principal plataforma para migrar bases de datos de terceros a una de tipo Oracle. Estas bases de datos externas son Access, SQL Server, Sybase ASE, DB2 y Teradata.

Además de este gran abanico de funcionalidades adaptadas a los requisitos de diferentes usuarios, existen otras ventajas destacables:

- Dentro de la comunidad de usuario, el hecho de presentar una interfaz gráfica sencilla, intuitiva y fácil de usar es la ventaja más repetida. Las opciones disponibles están agrupadas en menús consistentes y los iconos son representativos.

- Soporta conexiones tanto a bases de datos Oracle como a las no Oracle MySQL, Microsoft SQL Server, Microsoft Access, Sybase, Teradata e IBM DB2. La lista concreta de bases de datos para las que está certificada este software es visible en http://www.oracle.com/technetwork/developer-tools/sql-developer/certification-096745.html.

- Incluye autenticación de usuarios vía LDAP, Kerberos, sistemas de autenticación externa y proxy.

- Permite exportar datos en varios formatos comúnmente aceptados: xml, csv, sql insert, sql loader, txt, html y xls.

- Cuenta con una gran comunidad de expertos y manuales para ayudar al usuario, además de foros y vídeos (todos ellos tanto oficiales como no oficiales). Según la web oficial de la comunidad SQL Developer de Oracle, actualmente esta supera los 4 millones de usuarios (Oracle Community).

Por otro lado, caben mencionar también las siguientes limitaciones:

- Las utilidades disponibles en conexiones a bases de datos no Oracle están limitadas, aunque en todas se garantiza la navegación por los objetos y datos de la base de datos en cuestión.

- Los formatos aceptados para exportar e importar datos está muy desequilibrados a favor de la exportación. Sería útil ampliar los formatos de importación.

- Las funciones más avanzadas tienen restringido el uso por licencia, de manera que no puede considerarse como una herramienta completamente gratuita. Esta limitación afecta especialmente a las utilidades relacionadas con la optimización de consultas.

- Utilizarlo consume muchos recursos del equipo, de forma que conviene limitar el número de aplicaciones paralelas que se estén ejecutando para minimizar el riesgo de bloqueo o cierre inesperado. No recomendable en equipos con una memoria RAM limitada.

En conjunto, SQL Developer es una herramienta completa, actualizada y potente que merece la pena ser considerada a la hora de escoger una herramienta para gestionar bases de datos.

TOAD: Surgida inicialmente como una herramienta gratuita desarrollada por Quest Software y centrada en los desarrolladores, actualmente Toad se ha convertido en un estándar en lo que se refiere a gestores y desarrolladores de bases de datos. Su nombre refleja este objetivo inicial, y es acrónimo de 'Tool for Oracle Application Developers'.

El 28 de septiembre de 2.012 Dell confirmaba la adquisición de Quest Software, un proveedor de software centrado en desarrollar soluciones IT y con un amplio portfolio de productos que cubrían cuatro áreas: gestión de sistemas, seguridad, Business Intelligence y aplicaciones. Los productos de Quest y Dell pudieron acoplarse y complementarse, mejorando a Dell.

La principal fortaleza de Toad es haberse erigido como estándar dentro de los software dedicados a la gestión de bases de datos. Ello es visible en la alta puntuación que recibe de forma unánime en la comunidad de usuarios, la clara preferencia de estos (cuando se omite la restricción presupuestaria) y la extensa base de conocimiento formada por foros, documentos, vídeos y otras iniciativas formativas. Además, a pesar del traspaso de propiedad ocurrida en 2.012, el núcleo de desarrolladores de esta herramienta ha permanecido y muchos cuentan con una experiencia superior a los 10 años. En estos momentos, cuenta con más de 4 millones de visitantes, más de 450.000 usuarios registrados y más de 50.000 artículos formando su *knowledge base* (Toad World Official Page).

Inicialmente Toad fue desarrollado para ser compatible con Oracle (la primera versión fue publicada en 1.998), pero hoy en día se ha expandido a otras plataformas como SQL Server, MySQL y DB2 y en cada una de sus versiones ha mejorado o aumentado el número de capacidades disponibles.

Los hitos más importantes de su historia han sido:

- El primer lanzamiento tuvo lugar en octubre de 1.998 por parte de Quest. Sin embargo, comenzó a ser desarrollada en 1.995 por un único desarrollador.

- Su versión 8.5, publicada en julio de 2.005, incluyó soporte a la versión 10g de Oracle.

- La versión 10.0, de noviembre de 2.009, estuvo centrada en proveer estabilidad, usabilidad y consistencia. Esto motivó la estandarización de las

tablas de datos, el soporte a la codificación Unicode y a la herramienta de diseño de diagramas ER.

- La versión 10.5, de abril de 2.010, añadió más usabilidad al expandirse a más categorías de usuarios. Ya no eran sólo desarrolladores y administradores de bases de datos, también managers y analistas.

- La versión actual, 12.9, se han añadido mejoras como la posibilidad de filtrar los resultados y la monitorización de índices para ayudar a decidir si continúan o no siendo de utilidad.

Entre las características que hacen de Toad el gran referente que es destacan:

- La curva de aprendizaje es asequible para usuarios con poca experiencia en herramientas similares debido a una interfaz gráfica amigable con un alto grado de usabilidad.

- Permite personalizar el entorno de desarrollo, moviendo los paneles de funciones y trabajo a placer.

- Abarca un amplio espectro de funcionalidades que la convierten en una herramienta apta para múltiples fines y usuarios. Operaciones relacionadas con la administración, el desarrollo, la optimización de consultas, generación de diagramas ER, construcción de consultas, … Todas tienen cabida en Toad.

- La amplia base de conocimiento y el tamaño de una comunidad integrada tanto por expertos en la herramienta como principiantes y usuarios en formación. Muestra un gran énfasis en la formación.

En cambio, presenta también las siguientes limitaciones:

- Existen quejas de soporte especialmente en las versiones ajenas a Oracle, es decir, en versiones desarrolladas para otras plataformas.

- El tamaño del buffer es reducido, lo que provoca problemas a la hora de extraer cuantiosos datos en los resultados.

- No permite ejecutar múltiples consultas simultáneamente, sino que cada una debe ser lanzada desde una ventana distinta. Esta funcionalidad sí que está permitida por ejemplo en SQL Server Management Studio.

- El precio de las licencias.

- Responde lentamente cuando aumenta el número de conexiones.

Dentro del conjunto de herramientas de administración y desarrollo disponibles, Toad es indiscutiblemente la ganadora. Su largo historial, apoyo de la comunidad

internacional y flexibilidad para trabajar con múltiples plataformas han contribuido a mantenerla a la cabeza de las listas.

MyOra: es una herramienta completamente gratuita que permite monitorizar la ejecución de sentencias SQL cuya primera versión apareció en 2.009. Desde entonces, anualmente se ha publicado una nueva versión. En comparación con las herramientas presentadas anteriormente, MyOra no ofrece las mismas capacidades pero se ha decidido incluir en el estudio dada la utilidad que pueda tener académicamente si se emplea de forma complementaria a otras herramientas.

El atractivo de este software reside en la posibilidad de observar en tiempo real el rendimiento de la base de datos, una funcionalidad que en el caso de SQL Developer viene protegida por licencia. No sólo monitoriza el comportamiento en tiempo real, sino que también es capaz de monitorizar múltiples bases de datos simultáneamente, mostrar las sentencias SQL y las sesiones que más recursos consumen, así como filtrar el tipo de estadísticas que se desean observar (PIOs, LIOs, tiempo en CPU y tiempo pasado, o tipos de eventos que hayan tenido lugar).

Como características de despliegue esenciales destacan su única compatibilidad con el sistema operativo Windows, la posibilidad de soportar varias conexiones a una base de datos y la adopción del modelo MDI (Multiple Document Interface) para su interfaz gráfica.

Como su propio nombre indica, esta herramienta sólo puede ser empleada para analizar bases de datos Oracle, que soporta desde la versión 9i.

Así pues, aunque MyOra no pueda tenerse en contarse al mismo nivel que dbForge, Navicat, SQL Developer o TOAD, sí aporta una utilidad de forma gratuita que puede ser explotada si se cumplen los requisitos de uso en cuanto a plataforma de base de datos y sistema operativos empleados.

Finalmente y como resumen, se incluye a continuación la siguiente tabla comparativa entre las herramientas consideradas:

Característica	dbForge Studio	Navicat	SQL Developer	TOAD	MyOra
Licencia	Licencia gratuita temporal (30 días)	Licencia gratuita temporal (14 días)	Libre, extras bajo licencia	Licencia gratuita temporal (30 días)	Propietaria
Desarrollador	Devart	PremiumSoft CyberTech Ltd.	Oracle	Dell	Jayam Systems LLC
SO soportado	Windows	Windows, Linux, Mac OS	Windows, Linux, Mac OS	Windows	Windows
Compatibilidades	Oracle, MySQL	Oracle, MySQL, PostgreSQL, SQL Server, SQLite, MariaDB	Oracle, MySQL, SQL Server, Access, Sybase, Teradata, DB2	Oracle, SQL Server, MySQL, DB2	Oracle
Funciones	Todas contienen un asistente que permite crear y modificar bases de datos, tablas, procedimientos y disparadores, explorar estos elementos y además modificar la sintaxis de color para cada sección del código. La excepción es SQL Developer, que permite agregar o borrar conexiones a una base de datos, pero no crear una *per se*. Todas añaden un creador de consultas visuales y un diseñador de modelo visual o diagrama ER.				Sólo está dedicada a tareas de monitorización
Otros comentarios	Funcionalidades restringidas según edición (express, standard, profesional)		Respaldada por una gran empresa	Tiene una de la mayores comunidades de usuarios	No contiene pruebas temporales

Tabla 6: Comparativa de herramientas visuales

En base a lo anteriormente expuesto se han escogido para el estudio las herramientas SQL Developer de Oracle y Toad for Oracle de Dell, por sus compatibilidades, funcionalidades ofrecidas, y su fortaleza de imagen entre la comunidad de usuarios y administradores de bases de datos. En la sección 4.2 se llevará a cabo un análisis más detallado de estas dos herramientas, centrándose especialmente en su usabilidad a nivel de usuario y adecuación a los objetivos del proyecto. Indicar que MyOra no será incluida en esta comparativa, ya que las funcionalidades aportadas no son comparables con las de SQL Developer ni TOAD.

3.4 Discusión

El desarrollo que ha tenido las últimas décadas el sector de las tecnologías, y los mecanismos de almacenamiento y tratamiento de datos en concreto los han situado como una de las grandes ramas de la tecnología.

Acompañando este desarrollo del conocimiento también se densificó el tejido empresarial, y con él la variedad de productos a disposición del usuario. Oracle se encuentra entre las empresas más potentes del sector, lo que garantiza unos estándares de calidad y diseño de los que se benefician sus productos. La integración existente entre ellos y la cantidad de recursos relacionados con ellos junto a sus características técnicas revisadas anteriormente han motivado la elección de esta base de datos y su herramienta de gestión SQL Developer.

Por otro lado, se ha escogido TOAD, una herramienta que se ha convertido en la referencia dentro de los gestores de bases de datos. Con el fin de explorar su uso y probarla en primera persona, sobre unos ejemplos con distintas complejidades y operaciones, se decidió incluirla como segunda herramienta y que aquellos interesados en adquirir unas habilidades básicas puedan aprovecharse de este trabajo.

Las elecciones tomadas para acotar el alcance del proyecto están basadas esencialmente en la utilidad que puedan reportar a usuarios inexpertos actualmente desarrollando sus carreras universitarias, en la sencillez de uso y la curva de aprendizaje requerida. Esto explica por qué algunas medidas objetivas presentes en estudios de mercado a nivel de investigación o profesional como los tiempos de ejecución o rendimientos de las bases de datos no han sido tratados.

Las principales conclusiones extraídas del estudio, así como sus limitaciones y propuestas de mejora futura serán incluidas al final del documento.

4. Optimización de consultas: Mecanismos y herramientas

La cuarta sección del documento abarca los conceptos técnicos sobre los cuales se fundamenta el estudio. Estos han sido divididos a su vez en dos apartados. En él se explican los mecanismos y herramientas disponibles para medir el rendimiento de una sentencia SQL (en este caso consultas). Entre ellos hay incluidos algunos que si bien no son aplicables en este proyecto concreto, se ha considerado relevante tener conocimiento de ellos.

4.1 Mecanismos

La optimización de consultas en Oracle busca mejorar las consultas por tres vías distintas: 1) reduciendo la carga de trabajo, 2) balanceando la carga de trabajo, y 3) paralelizando la carga de trabajo. La primera opción conlleva un reanálisis de la consulta para hacerla más eficiente. La segunda está especialmente indicada para entornos reales corporativos en los que la ejecución de algunas tareas pueda relegarse a horas en los que la carga de trabajo sea menor, por ejemplo horario nocturno. Entre estas se encontrarían las actualizaciones de los sistemas y la generación de informes de uso, de seguridad, etc. Por último, la tercera requeriría otro tipo de sistema en el que fuera posible paralelizar el trabajo, y el entorno de pruebas de este estudio no lo permite, ya sea por incompatibilidad con máquinas virtuales o por la limitación que tiene de uso de un solo procesador. Así pues, la mejora de las consultas se centrará en la primera opción, y en concreto se valdrá de los siguientes mecanismos y herramientas (apartados 4.1 y 4.2):

4.1.1 Paquete DBMS_XPLAN: Explain Plan For, Display y Display_Cursor

El paquete dbms_xplan fue introducido por primera vez en la versión Oracle 9i, y a lo largo de las siguientes versiones ha incrementado el número de funciones dedicadas a mostrar explain plans desde diversas fuentes: el comando 'explain plan', la vista v$sql_plan, el repositorio AWR ('Automatic Workload Repository'), el SQL Tuning Set (STS) y el SQL Plan Baseline (SPM). Para información más detallada se recomienda consultar la guía oficial de administración de Oracle 11g r2.

Este paquete permite mostrar uno de los posibles planes de ejecución ideados por el optimizador de Oracle para ejecutar una determinada sentencia, pero que no llega a ejecutar en dicho momento. Por eso, se trata de una estimación que realiza Oracle, a partir de estadísticas que ya tuviera sobre los objetos involucrados e información específica contenida en la versión de optimizador que se esté usando. Este último puede comprobarse consultando el valor del parámetro '`optimizer_features_enable`' y en este caso es la 11.2.0.2.

Utilizar el *explain plan for* para analizar el comportamiento de una consulta es útil porque muestra la siguiente información:

- Las tablas referenciadas en la consulta, los índices involucrados y el método de acceso ('*access path*'*) elegido para cada uno de estos objetos.
- Información de los join: tipo (outer, inner, semi, etc.), método (hash, sort-merge, etc.) y orden (qué tabla se une a cuál).
- Operaciones de filtrado, ordenación o agregación.
- Coste en tiempo y recursos de la CPU en total y por cada operación. El valor del parámetro CPU no es manipulable para mejorar el rendimiento de la consulta, sino que es asignado internamente por el sistema en función de las estadísticas y de las estimaciones que se hagan sobre entradas y salidas, uso de la CPU y recursos de la red que serán requeridos por la consulta. Esto permite comparar los explain plan entre sí, y es criterio de elección de plan de ejecución para el optimizador. Cuanto más bajo sea el coste de un explain plan, más eficiente se espera que sea.
- Cardinalidad de cada operación ejecutada: número de líneas estimadas que devolverá una operación. La exactitud de estimación de este parámetro es vital para generar un explain plan adaptado a la realidad, porque en función de esta estimación el optimizador decidirá el resto de parámetros del plan, pudiendo resultar en un plan subóptimo. Para mejorar esta estimación puede recurrirse a las estadísticas, que se presentarán próximamente.
- Si procede, información de particiones y ejecución paralela. Este último punto de información no es relevante por las características de la versión Oracle y la base de datos empleadas.

Por otro lado, existen limitaciones a tener en cuenta:

- Los explain plan que pueda generar el optimizador pueden variar debido a 1) que el esquema desde el cual se generen y el esquema en el que realmente se ejecute la consulta no sean los mismos, 2) el objetivo de costes (tiempo, número de I/O) cambie, 3) la versión de Oracle que se esté utilizando, y 4) el estado en que se encuentre el sistema en el momento de ejecutar la consulta (en cuanto a capacidad de memoria, número de procesos, recursos disponibles, etc.).
- Cuando hay variables *bind* involucradas, no son fiables porque no las tienen en cuenta (*bind peeking*).
- Generar un explain plan requiere aplicar un '*parse*'*, lo que consume más recursos.

- El *explain plan* mostrado puede no ser el plan de ejecución final, por lo que el análisis estará incompleto hasta que se extraiga también este último. Es decir, para hacer un diagnóstico real de la ejecución de una consulta hay que centrar el análisis en las instrucciones que finalmente se han seguido, no las que se tenía planeado seguir.

Respecto a este último punto, Oracle ofrece cuatro mecanismos: Autotrace, SQL Monitor, TKPROF y DBMS XPLAN. Todos ellos se verán en las siguientes subsecciones.

La información de los *explain plan* viene contenida en una tabla llamada "plan_table", generada automáticamente por el script de Oracle "utlxplan.sql" (en C:\oraclexe\app\oracle\product\11.2.0\server\rdbms\admin\utlxplan.sql, que se corresponde con la ruta @ORACLE_HOME/rdbms/admin). Pueden consultarse los campos de esta tabla mediante el comando '`desc plan_table;`'.

Los comandos "`explain plan for`" y "`select plant_table_output from table (dbms_xplan.display());`" se utilizan para generar y mostrar respectivamente el último explain plan de la consulta que sigue al primer comando. El segundo llama a la función `dbms_xplan.display`, que pertenece al paquete `dbms_xplan` introducido por primera vez en la versión Oracle9i R2.

La función `display()` puede ser llamada especificando más información, como el nivel de detalle de la salida: básico, típico, serial o completo. En este estudio se trabajará siempre con la tabla por defecto "*plan_table*" para guardar los *explain plan* y sobre la última sentencia insertada en la *plan_table*, de forma que las consultas de dbms_xplan.display tendrán la forma "`select * from table (dbms_xplan.display('plan_table', null, ['basic'|| 'typical' || 'serial' || 'all']));`".

Otra alternativa es utilizar la función "`dbms_xplan.display_cursor ([SQL_id],[Child number], [Format]);`". Este permite consultar el plan de ejecución tanto de la última sentencia (SQL_id = null) como de otra anterior (SQL_id != null), siempre que el cursor en cuestión haya sido cargado en la caché de cursores, y definir con más detalle el formato. Por ejemplo, el comando "`dbms_xplan.display_cursor([null || 'sql_id'],null,'all');`" mostraría el plan de ejecución de la última sentencia ejecutada en la sesión actual.

El acceso a esta función del paquete dbms_xplan requiere los privilegios 'execute' sobre el paquete dbms_xplan (`grant execute on dbms_xplan to hr;`) y 'select' sobre las vistas dinámicas `v$sql_plan`, `v$session` y `v$sql_plan_statistics_all`.

Si a esta última función se le añade el hint "`gather_plan_statistics`" o se establece el nivel completo de estadísticas (`statistics_level = 'all'`), mostrará también los valores estimados y los reales sobre cardinalidad, el tiempo real invertido

en cada operación,y el número de LIOs* y PIOs*, todo ello extraído de la vista `v$sql_plan_statistics_all`.

El formato acepta modificaciones para personalizar la salida del comando más allá de los formatos generales (`basic, typical, all`). Estas palabras clave son: `rows, bytes, cost, partition, parallel, predicate, projection, alias, remote, note, iostats, memstats, allstats` (que es equivalente a '`iostats memstats`') y `last`. Es importante tener en cuenta que '`iostats`' requiere haber recopilado estadísticas (ya sea a través del hint `gather_plan_statistics` o por el parámetro a nivel de sesión `statistics_level = all`) y para '`memstats`' el valor del parámetro '`pga_aggregate_target`' debe estar activado (esto es, tener un valor distinto de 0). Para más información se recomienda consultar la guía de paquetes de Oracle.

Activar las estadísticas de memoria (`memstats`) añade las columnas OMem, 1Mem, UsedMem y UsedTmp. OMem representa 'Estimated_Optimal_Sizenumber', es decir, el tamaño del área de trabajo SQL que se estima que requerirá la operación para ser ejecutada completamente en memoria (ejecución óptima). 1Mem representa 'Estimated_Onepass_Size', el tamaño estimado que requeriría el área de trabajo para ejecutar la operación en una transferencia (pass). Used-Mem es la memoria usada por el área de trabajo durante la última ejecución del cursor que se esté analizando. Finalmente Used-Tmp es el espacio usado en temp para la última ejecución. v$sql_workarea

Por su parte, las estadísticas `iostats` añaden las columnas Reads, Buffer y Writes. Los 'Read' representan lecturas físicas, y los 'Buffer' lecturas lógicas. Si la sentencia fuera de tipo 'insert' o 'update', aparecería una columna adicional llamada 'Writes' para las escrituras.

`Display_cursor` es especialmente útil cuando se emplean variables *bind*, para comprobar si el explain plan generado en '*Autotrace*' y el plan de ejecución son iguales o no.

El explain plan puede venir acompañado de una sección de notas ('`note`') con información adicional, como si se ha utilizado muestreo dinámico y dónde, si se ha empleado el formato típico o completo (`all`).

4.1.2 Plan de ejecución ('Execution Plan')

Es la secuencia de instrucciones que Oracle finalmente lleva a cabo al ejecutar una consulta, y puede variar respecto a la predicción que se hizo en los *explain plan*.

Consultar cuál ha sido el plan definitivo ayuda a detectar qué áreas son las que más se han desviado de las predicciones, y por tanto dónde sería necesario aplicar modificaciones para obtener en el futuro estimaciones más exactas.

Para acceder a él hay disponibles varios métodos que se presentan en la sección 4 del proyecto, destacando: 1) la función '*autotrace*' (apartado 4.1.4), 2) la función dbms_xplan.display_cursor(), 3) trace en conjunto con TKPROF y 4) vista v$sql_plan.

El plan mostrado contendrá los siguientes parámetros, que serán repetidos en otros mecanismos:

- Plan Hash Value: identificador del plan.
- Id de la operación, empezando por 0. Dado que los tres casos de prueba son consultas, y por tanto comienzan con 'select', la primera operación siempre será 'select statement'.
- Nombre de la operación. Cada entrada aparecerá identada, para indicar en qué orden ha sido ejecutada cada acción. Estas pueden ser operaciones de acceso, procesamiento, selección, ordenación y filtrado de lo datos. La lectura del plan de ejecución es la misma que para los explain plan.
- Nombre del objeto involucrado, cuando la operación es de acceso. Oracle contempla seis métodos de acceso ('*access paths*'): *full table scan* (lectura complete de la tabla), *rowid scans* (escaneado mediante identificador de línea), *index scans* (escaneado mediante índice), *cluster access*, *hash access*, y *sample table scan* (escaneado mediante muestra de una tabla). Para obtener más información acerca de cada uno de estos métodos, se recomienda acudir a la guía oficial de Oracle para Tuning, capítulo 11.
- Número de líneas recuperadas ('*rows*') o cardinalidad.
- Cantidad de información, medida en bytes.
- Coste y porcentaje de CPU. Es uno de los criterios utilizados por el optimizador para escoger plan, y no es necesariamente mejor cuanto menor sea el número, por lo que conviene conocer si el entorno en el que se ejecutan las consultas prima el tiempo de ejecución o el coste de los recursos en términos de operaciones de entrada y salida.
- Tiempo dedicado a ejecutar la operación correspondiente.

Según el método utilizado para extraer el plan de ejecución y el nivel de detalle de las estadísticas podrán verse más detalles del plan. En general, se tendría:

- Función `dbms_xplan.display_cursor`: muestra el plan de ejecución de una sentencia identificada por su sql_id, y puede comparar valores estimados (E-*) con reales (A-*) en cada operación.

```
SQL> select * from table(dbms_xplan.display_cursor('0p9r7arfxdnjf',0,'all allstats last'));

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------------------------------------------------------------------
SQL_ID  0p9r7arfxdnjf, child number 0
-------------------------------------
select /* PC6 */ distinct count(employees.employee_id),departments.depar
tment_id from departments,employees where
departments.department_id=employees.department_id group by
departments.department_id order by departments.department_id

Plan hash value: 2741899567

------------------------------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation            | Name             | Starts | E-Rows |E-Bytes| Cost (%CPU)| E-Time   | A-Rows |   A-Time   | Buffers | Reads  |  OMem |  1Mem | Used-Mem |

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |                  |      1 |        |       |     7 (100)|          |     62 |00:00:00.05 |      24 |     19 |       |       |          |
|   1 |  SORT GROUP BY       |                  |      1 |     62 |   186 |     7  (15)| 00:00:01 |     62 |00:00:00.05 |      24 |     19 |  4096 |  4096 | 4096  (0)|
|*  2 |   INDEX FAST FULL SCAN| EMP_DEPARTMENT_IX |      1 |   4902 | 14706 |     6   (0)| 00:00:01 |   4910 |00:00:00.05 |      24 |     19 |       |       |          |
------------------------------------------------------------------------------------------------------------------------------------------------------
```

Ilustración 7: Ejemplo salida de dbms_xplan

- Función `autotrace`: muestra el plan de ejecución, pero puede no ser fiable al basarse en explain plans. Las estadísticas son en conjunto de toda la sentencia, no específica de cada operación.

```
Execution Plan
----------------------------------------------------------
Plan hash value: 2741899567

-------------------------------------------------------------------------------------------
| Id  | Operation            | Name             | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |                  |    62 |   186 |     7  (15)| 00:00:01 |
|   1 |  SORT GROUP BY       |                  |    62 |   186 |     7  (15)| 00:00:01 |
|*  2 |   INDEX FAST FULL SCAN| EMP_DEPARTMENT_IX |  4902 | 14706 |     6   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - filter("EMPLOYEES"."DEPARTMENT_ID" IS NOT NULL)


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
         24  consistent gets
         19  physical reads
          0  redo size
       1960  bytes sent via SQL*Net to client
        567  bytes received via SQL*Net from client
          6  SQL*Net roundtrips to/from client
          1  sorts (memory)
          0  sorts (disk)
         62  rows processed
```

Ilustración 8: Ejemplo salida autotrace (SQL*PLUS ommand line)

- Función `session_trace + TKPROF`: muestra también estadísticas de eventos y esperas, valores específicos para cada operación y tiempo real en CPU frente a tiempo total pasado.

- Vista dinámica `v$mystats`: muestra las estadísticas detalladas de la última ejecución, como el uso de CPU, el número de ordenaciones (‘sorts’), el número de llamadas de usuario, etc.

4.1.3 Estadísticas

Como se mencionó anteriormente en la sección del explain plan, contar con unas estadísticas actualizadas de los objetos involucrados en las consultas mejora las posibilidades de obtener un explain plan óptimo.

Existen dos formas de recabar información estadística sobre objetos de la base de datos, y son el comando "*analyze*" y el paquete "*dbms_stats*". Según el uso al que se destine la información, su aplicación varía. Si los datos obtenidos están destinados a ser utilizados por el optimizador para mejorar la ejecución de las consultas en las que el objeto esté involucrado, entonces lo recomendable es utilizar el paquete "*dbms_stats*". Si en cambio los datos obtenidos no son empleados por el optimizador como bloques vacíos o espacio empleado, "*analyze*" es mejor opción.

Tanto "*dbms_stats*" como "*analyze*" almacenan las estadísticas generadas sobre el almacenamiento físico de los objetos en el Diccionario de Datos ('*data dictionary*'), por lo que el uso de este mecanismo estará reservado a usuarios que tengan entre sus privilegios el acceso a dicho diccionario (`select any dictionary`). El paquete *dbms_stats* permite almacenar las estadísticas en tablas externas al diccionario de datos, de manera que el optimizador no se ve afectado y el usuario puede recoger todas las estadísticas que considera necesarias sin ver comprometido el funcionamiento del optimizador. Es decir, sólo las estadísticas almacenadas en el diccionario de datos influyen en el optimizador.

La recolección de estadísticas puede forzarse ejecutando una de las funciones del paquete, como `dbms_stats.gather_table_stats`, o incluyendo un hint en la propia consulta. Sin embargo, la función que recoge todas las estadísticas relativas a un mismo esquema es la función `dbms_stats.gather_schema_stats`.

Por último, el parámetro de inicialización "`statistics_level`" fijado en 'all' para la sesión actual sirve como alternativa al hint '`gather_plan_statistics`', y es indispensable para ejecutar correctamente funciones como '`dbms_xplan.display_cursor`' (sólo así aparecerán los valores reales de la ejecución). La elección de un método u otro puede afectar a los datos mostrados en la salida de display_cursor: los valores de A-time no son acumulativos si se utiliza el hint; pero si se utiliza el parámetro de inicialización se introduce un retardo extra debido a la granularidad de las estadísticas. Siguiendo las recomendaciones de Oracle, se recurrirá a los hint en última instancia.

4.1.4 Autotrace

La funcionalidad 'autotrace' en SQL*Plus es la que permite ver tanto el plan de ejecución seguido por Oracle en una consulta como las estadísticas ligadas a dicha ejecución, por lo que a diferencia del Explain Plan, aquí sí que se ejecuta la consulta. Para ello es necesario cumplir dos requisitos: 1) el usuario que la use debe contar con el

rol de usuario “PLUSTRACE”, y 2) la tabla “plan_table” debe aparecer en el esquema de dicho usuario.

Para activarlo en una sesión, basta con introducir “set autotrace on”. Esto mostrará tanto el plan de ejecución (ver apartado 4.1.2) como las estadísticas, pero es posible limitar la salida a sólo el plan o sólo las estadísticas. En este caso, se ha decidido mostrar ambas.

El número de llamadas recursivas, lecturas totales, lecturas físicas, ordenaciones en memoria y en disco, así como el número de líneas son los parámetros que más utilidad tienen al valorar un plan de ejecución.

- Las llamadas recursivas (‘*recursive calls*’) son las llamadas adicionales que Oracle realiza a otros objetos para poder ejecutar la consulta inicial introducida. La compilación (‘*parse*’) que se hace la primera vez que se ejecuta una consulta hace este valor sea distinto de cero la primera vez, pero no las siguientes, porque la información necesaria para ejecutar la consulta permanece en caché.

- ‘*Consistent gets*’ o lecturas de bloques que Oracle debe realizar, ya sea desde caché o desde disco. Cuanto menor sea este número, mejor.

- Lecturas físicas (‘*Physical reads*’): número de bloques que ha sido necesario traer desde disco para leerlos, por lo que cuanto menor sea este indicador, mejor ejecución. Al igual que ocurre con las llamadas recursivas, la primera vez que se ejecute la consulta este parámetro tendrá un valor más alto. Después es de esperar que el número decrezca hasta 0, gracias a la información que se almacene en caché, aunque depende del tamaño de la tabla y los datos extraídos. Puede tomarse como una medida del número de entradas y salidas llevadas a cabo para recuperar bloques de información.

- Ordenaciones (‘*sorts*’): según la dificultad de la consulta y el tamaño de la base de datos, esta operación puede afectar gravemente al tiempo de ejecución de la consulta, por lo que conviene reducir lo máximo posible su uso, en concreto las ordenaciones en disco.

- ‘*Rows processed*’ indica cuántas filas forman el resultado de la consulta, aunque en realidad para llegar a él hayan sido procesadas más.

Aunque esta herramienta tiene a su favor la facilidad de uso, tiene dos importantes limitaciones: 1) no muestra las estadísticas por cada fase del plan, y 2) es posible que el resultado no sea exactamente lo que Oracle haya ejecutado porque emplea explain plans.

Como alternativas a esta función se tiene la función Autotrace en SQL Developer, SQL Monitor, TKPROF y DBMS_XPlan. La primera será tratada en el apartado dedicado a SQL Developer. La segunda opción no requiere esperar a que la consulta termine de ejecutarse para visualizar las estadísticas, ya que muestra la ejecución en tiempo real. Sin embargo su uso está restringido por licencia del Tuning Pack, por lo

que queda fuera del alcance de este estudio. Las opciones restantes serán tratadas en las siguientes subsecciones.

4.1.5 Paquete DBMS_MONITOR - Función SQL Trace

Esta función recoge información estadística de las ejecuciones que tengan lugar desde que se activa hasta que se desactiva, por lo que no está limitado a mostrar la información exclusivamente de una consulta SQL. Incluso puede emplearse para recabar información de otras sesiones distintas, lo que la convierte en una herramienta potente.

El comando "`alter session set sql_trace = true`" sería el método de activación más sencillo, pero es más recomendable utilizar el método `dbms_monitor.session_trace_enable`, centrado exclusivamente en la sesión actual. Consta de cinco parámetros en los que se puede especificar más qué estadísticas recoger. Para más información de los comandos, consultar los anexos.

Rastrear la ejecución de operaciones SQL conlleva una sobrecarga en el sistema, por lo que se desaconseja mantenerla activa más tiempo del necesario o utilizarla para rastrear toda la base de datos. Esto explica la granularidad que puede alcanzarse manejando correctamente los métodos de dbms_monitor.

Toda la información de las estadísticas recogida por SQL Tracing se almacena en un fichero llamado '*trace file*', alojado en la ruta de 'user_dump_dest' y cuyo tamaño no podrá superar el marcado por 'max_dump_file_size'.

4.1.6 Paquete DBMS_MONITOR: Función TKPROF

Es una funcionalidad complementaria a sql_trace, que permite generar informes más fáciles de interpretar a partir de los ficheros de trazas ('*trace file*') sobre la ejecución de una consulta determinada. Requiere que el usuario que lo lance cuente con el privilegio "execute" para el paquete "dbms_monitor", y su concesión recae en el usuario administrador.

Los datos que recopila SQL Tracing de la ejecución son volcados en el archivo 'trace file' almacenado en la ruta del parámetro "user_dump_dest" y son tratados por TKPROF para generar el informe. Para poder comparar los informes de cada ejecución, puede cambiarse el nombre del archivo de trazas sobre el que se genera el informe TKPROF.

Su uso requiere en primer lugar especificar el trace file en el que se quiera almacenar la información para separar los resultados de las pruebas (`alter session set tracefile_identifier='prueba000';`), y a continuación activar tanto la recolección de estadísticas (`set timed_statistis = true;`) como la funcionalidad

SQL trace (`execute dbms_session.set_sql_trace(true);` o `execute dbms_monitor.session_trace_enable(waits => TRUE, binds => TRUE);`). Después de ejecutar la consulta (incluyendo un '`commit`') y desactivar la trazabilidad para ahorrar recursos (`execute dbms_session.set_sql_trace(false);`o `execute dbms_monitor.session_trace_disable;`), ya aparecería en el directorio de los archivos de trazas el nuevo, con extensión ".trc".

Saliendo de la consola SQL y situándose en el directorio especificado en "user_dump_dest", que en este caso ha sido "C:\oraclexe\app\oracle\diag\rdbms\xe\xe\trace", se ejecuta el siguiente comando: "`C:\oraclexe\app\oracle\diag\rdbms\xe\xe\trace>tkprof prueba000.trc pruebaTKPROF.prf`", y se obtiene el informe TKPROF "pruebaTKPROF.prf".

En él se distinguen secciones diferenciadas:

- Introducción: la versión del tkprof con el que se ha generado el informe, la fecha y hora, el nombre del archivo de trazas origen y las opciones de tkprof, si las hubiera. Estas opciones sort pueden ser: default, A continuación se incluye una descripción de los parámetros sobre los que se han extraído los datos ('count', 'cpu', 'elapsed', 'disk', 'query', 'current' y 'rows').

- Por cada sentencia ejecutada, muestra una tabla con los valores de los parámetros según se trate de operación 'parse', 'execute' o 'fetch'. Además indica que se ha encontrado o no un plan previo en la caché, el modo del optimizador, el id del usuario que lo ha ejecutado y si procede, el número de estadísticas capturadas. Este informe también incluye el plan de ejecución, que aquí muestra por cada operación otros datos como el número de lecturas físicas y lógicas, el tiempo que ha dedicado a esta operación, el coste que ha tenido y el número de bytes (tamaño) de los datos manejados.

- Opcionalmente, si así se indicó en la opciones de trazabilidad, se añadiría en la anterior sección por cada sentencia un apartado dedicado a las esperas ('waits'). En él se enumeran los eventos que han podido provocar una espera en la ejecución, el número de veces que se ha repetido un evento concreto, el máximo tiempo que se ha esperado por dicho evento y el tiempo total esperado por ese evento en la ejecución de la sentencia que se esté analizando.

4.1.7 Paquete DBMS_PROFILER

Es un paquete de funciones desarrollado por Oracle para recabar información sobre la ejecución de líneas de código para detectar cuellos de botella. El objetivo es obtener una mejora en el rendimiento. Utilizar este paquete requiere que el usuario tenga el privilegio 'CREATE', no sólo 'EXECUTE ONLY'. De otra forma, no se mostrarán los resultados.

Conocer su existencia es relevante porque forma parte del grupo de mecanismos que Oracle pone a disposición de los usuarios para optimizar el rendimiento, aunque en este escenario no sea posible aplicarlo ya que sólo es aplicable a procedimientos y funciones.

4.1.8 Paquete DBMS_MONITOR

Como se ha visto en las secciones 4.1.4 y 4.1.5, las funcionalidades SQL Trace y TKPROF están incluidas en él. Es el paquete de funcionalidades de Oracle dedicado a tareas de monitorización de la base de datos. COMPLETAR

4.1.9 Paquete DBMS_RESULT_CACHE

Oracle incorpora en la versión 11g el paquete DBMS_RESULT_CACHE, que permite al administrador de la base de datos (o el usuario que tenga los privilegios correspondientes) gestionar la parte de la Shared Pool usada por la caché de resultados SQL y PL/SQL. Esta funcionalidad sin embargo no está incluida en la edición exprés, pero sí en la Enterprise, y dada su novedad y utilidad ha sido incluida en el proyecto.

La caché de resultados contiene los resultados finales de una ejecución de una consulta, de forma que permite ser reutilizado por otros usuarios que también accedan al shared pool. La incorporación de esta característica ofrece grandes ventajas en entornos en los que los sets de resultados no cambian con mucha frecuencia, al permitir accesos de alta velocidad a los mismos.

Existen dos modos para esta caché: 'manual' o 'force'. En modo 'manual', sólo se utilizará la caché de resultados si así lo indica la consulta, es decir, mediante hints. En modo 'force' ocurre lo opuesto, por defecto busca en la caché de resultados, salvo si se indica explícitamente con el hint 'no_result_cache' que no lo haga. El modo por defecto es manual.

Dentro de este paquete se incluyen las funciones DBMS_RESULT_CACHE.BYPASS y DBMS_RESULT_CACHE.FLUSH. La primera configura el modo de uso de las cachés y su entorno de aplicación (sólo a la sesión desde la que se llama a la función o todas las sesiones), mientras que la segunda borra el contenido de esta caché e incluso las estadísticas relacionadas. Conocer con más detalle el estado de esta caché y sus estadísticas es posible utilizando la función DBMS_RESULT_CACHE.MEMORY_REPORT, que requiere haber activado previamente "`set serveroutput on;`".

Al igual que sucedía con el paquete DBMS_MONITOR, el usuario administrador del sistema debe otorgar el privilegio 'execute' a los usuarios que vayan a llamar a estas funciones: `SQL> grant execute on dbms_result_cache to (usuario);`.

4.1.10 Hints u órdenes de ejecución:

Son instrucciones que se envían al optimizador para forzar una determinada forma de actuar. Su uso suele estar poco recomendado, ya que se espera que el optimizador cuente con información más detallada que el usuario y su toma de decisiones para generar un plan de ejecución sea más adecuada. Por ello, se emplean como último recurso, y sólo cuando las estadísticas que utiliza el optimizador en su toma de decisión no están actualizadas.

Oracle divide los hints en nueve categorías: 1) hints para modificar el objetivo de optimización, 2) para habilitar características de optimización, 3) hints de métodos de acceso, 4) hints para ordenaciones en joins, 5) hints para operaciones en joins, 6) hints para la actualización de aplicaciones online, 7) hints para ejecución paralela, 8) hints para transformaciones de consultas y 9) otros hints. [Oracle DB Performance Tuning Guide, capítulo 19]

De todos ellos, se presentan a continuación los hints más destacables:

- "*Result Cache*": Obligando a acudir en primer lugar a esta área, es posible que se reduzcan las lecturas de disco si la consulta es repetida recurrentemente. Sin embargo, para que sea ventajoso su uso, debe conocerse la popularidad de la consulta y si los datos son actuales. Para activarlo bastaría con introducir "/*+ RESULT CACHE */" en la consulta o cambiar el valor del parámetro "*result_cache_mode*" a "auto". Esta segunda opción mueve todos los resultados de la consulta a la cache de resultados. Para borrarla, se ejecuta el procedimiento "*dbms_result_cache.flush*". Es una funcionalidad reciente, de Oracle DB 11g R1, pero no incluida en la edición exprés.
- "*Gather_Plan_Statistics*": obliga a almacenar la cardinalidad real de cada operación aplicada en una sentencia, y en conjunto con la función dbms_xplan.display_cursor (format => 'allstats last') muestra tanto la cardinalidad estimada (E-Rows, estimated - rows) como la real (A-Rows, actual - rows).

4.1.11 Variables 'bind'

Las variables bind son variables creadas en SQL*Plus a las que se hace referencia en funciones y procedimientos PL/SQL. Su principal utilidad radica en reducir el número de compilaciones ('*parses*') al realizar una consulta.

En una misma consulta en la que varía el valor de un parámetro (por ejemplo, el valor del department_id por el que se busca una información), si el valor del department_id se especifica claramente en la propia consulta, Oracle interpretará como dos consultas diferentes la que utilice el department_id 570 y la que use el 430. Esto conllevaría dos compilaciones de una consulta que en realidad es la misma, lo que lleva a consumir más recursos de los necesarios. Empleando una variable bind externa, la

consulta sólo se compilaría una vez y durante la ejecución extraería el valor concreto de dicha variable.

Como contrapartida, existe un problema llamado "*bind peeking*" que limita el uso de este mecanismo. Cuando la distribución de valores para un parámetro es muy desigual, un mismo plan puede no ser válido para todos ellos, lo que en ocasiones conduciría a un plan por debajo del óptimo. Por eso, sólo están recomendadas para parámetros en los que la selectividad es similar en todos sus valores.

Para tratar de paliar este problema Oracle 11g introdujo una nueva aplicación de los cursores adaptados ('*adaptive cursors*'), que son activados y usados de forma automática. Ahora serían capaces de decidir de forma inteligente si un plan debe ser recalculado o no, según los valores que hayan sido introducidos en las sucesivas ejecuciones. Si fuera así, el cursor se marcaría como "Bind-Sensitive" (sensible a la variable bind). Esta información puede consultarse en las columnas añadidas a la vista v$sql "`is_bind_sensitive`" e "`is_bind_aware`".

Si los diferentes valores del parámetro pueden alterar potencialmente el plan, el cursor es marcado como "Bind-Sensitive", y en la columna "`is_bind_sensitive`" aparecerá una 'Y'. Después de unas pocas ejecuciones, la base de datos cuenta con más información sobre el cursor y los valores, y decide si finalmente el cursor debe cambiar de plan según el valor del parámetro o no. Si esto se diera, el cursor es marcado como "Bind-Aware", y en la columna "`is_bind_aware`" aparecería un 'Y'. Por lo tanto, los cursores "Bind-Sensitive" son potenciales candidatos a ser cursores para los que el plan cambie con el valor del parámetro, mientras que los "Bind-Aware" han pasado a convertirse en ese tipo de cursores.

Sabiendo el sql_id de una consulta, se puede consultar información acerca de las variables bind consultando la vista `v$sql_bind_capture`.

4.1.12 Tablas, vistas y comandos de consulta útiles

Tabla `dba_roles`: muestra todos los roles que existen en la base de datos. Entre ellos están los roles "connect", "gather_system_statistics" y "plustrace", mencionados a lo largo del estudio.

Tabla `plan_table`: tabla en la que por defecto se almacena el explain plan de una consulta.

Vistas `session_privs` y `session_roles` del diccionario de datos: muestran respectivamente los privilegios y los roles que tiene actualmente asignado el usuario en la sesión activa.

Vistas `dba_sys_privs` y `dba_role_privs`: muestran los privilegios de sistema y roles disponibles, junto al poseedor de dicho privilegio o rol y si tiene capacidad para

otorgarlo a un tercero. Si se utilizan las vistas homónimas a user_*, se mostrará esta información para el usuario actual. La vista `role_tab_privs` muestra información de los roles disponibles.

Vista `v$sql_plan`: vista dinámica de ejecución introducida en Oracle 9i que muestra el plan de ejecución para una sentencia SQL que ha sido compilada en un cursor y almacenada en la caché del cursor. El resultado de esta vista puede no coincidir con el del explain plan, entre otros motivos porque el plan mostrado en esta vista sí tiene en cuenta las variables bind.

"`Show parameters`": permite consultar la configuración actual de la base de datos. Son especialmente importantes los parámetros relacionados con el optimizador, las estadísticas, los modos de la caché y trazabilidad. El significado de cada uno de estos parámetros puede consultarse mediante "`select name, description, value from v$parameter`".

Vista `v$result_cache_objects`: Para monitorizar la caché de resultados. Muestra los objetos relacionados con esta cache. Dado un "cache_id" en un plan de ejecución, puede realizarse la siguiente consulta sobre la vista: "`select id, type, creation_timestamp, block_count, row_count from v$result_cache_objects where cache_id='…';`".

Vista `v$sql_bind_capture`: muestra información de las variables bind utilizadas por los cursores SQL. Cada línea de la vista contiene información sobre una determinada bind, siempre y cuando el parámetro de inicialización "statistics_level" no sea 'basic' (para este nivel la captura de bind está desactivada).

Vista `v$sql_plan_statistics_all`: vista utilizada por la función dbms_xplan.display_cursor para mostrar estadísticas sobre la ejecución (no estimación) de una consulta.

Vistas `v$mystats` y `v$statname`: la segunda muestra el id de las 612 estadísticas disponibles en la vista, y la primera el valor de cada una de esas estadísticas para la última ejecución. La herramienta Autotrace de SQL Developer extrae los datos de estas vistas, por eso para utilizarlo se requieren permisos sobre ellas (ver anexos). Hay otros ID en las vistas v$sesstat y v$sysstat que corresponden a otras estadísticas. Los ID asignados a cada estadística pueden variar entre versiones de Oracle.

Vista `v$session`: permite consultar el número SID, sql_id, nombre del usuario, estado de la conexión y módulo de la conexión (qué medio se ha utilizado para crear esa conexión, como consola o SQL Developer).

Vista `v$sql_monitor`: Incluye estadísticas sobre la ejecución de una sentencia (`v$sql_monitor.*`), entre las que destacan `elapsed_time`, `cpu_time`,`fetches`, `buffer_gets`, `disk_reads`, y `user_io_wait_time`. Lamentablemente, la disponibilidad de esta vista está limitada por la licencia de Oracle para el paquete 'Oracle Diagnostic and Tuning Pack'.

4.2 Herramientas

El desarrollo de este estudio se fundamenta en el uso de cuatro herramientas principales: 1) Oracle DB XE 11g Release 2 y su optimizador interno, 2) SQL Developer y 3) Toad.

4.2.1 Oracle DB XE 11g R2

Oracle DB XE 11g r2 es el RDBMS elegido frente a otras opciones como MariaDB o SQL Server, como se explicó en el apartado 3.1. El nombre completo de este sistema es Oracle Database Express Edition 11g Release 2, disponible desde 2014. Este software es gratuito, y está disponible para los sistemas operativos más comunes: Windows, Linux y Mac OS. De estos tres, se ha empleado el adaptado a Windows de 64 bits. Frente a su antecesor, la versión 10g, ofrece hasta 11GB de almacenamiento de datos de usuario (la 10g sólo 4GB).

En virtud del acuerdo de licencia de la versión escogida, se han detectado las siguientes limitaciones que afectan al desarrollo del proyecto:

- No incluye las funciones “PL/SQL Function Result Cache” ni “Query Results Cache”.
- No incluye las funciones “Oracle Diagnostic Pack”, “Oracle Tuning Pack” ni “SQL Plan Management”. Además, las dos primeras son funcionalidades que sólo están disponibles bajo licencia, por lo que quedarían igualmente excluidas del estudio.
- Paralelismo, debido a su restricción de utilizar un único procesador aun habiendo disponibles más.

A pesar de estas limitaciones, la edición exprés aplicada a Windows sigue ofreciendo unas características atractivas para el entorno de trabajo en el que se desarrolla este proyecto, que le convirtieron en el rdbms escogido para llevarlo a cabo. En concreto, destacan la compatibilidad con el lenguaje SQL, la aceptación de las funciones y procedimientos PL/SQL, y la conservación de las funciones analíticas en SQL.

Cabe indicar que en esta versión de Oracle se han introducido mejoras que afectan directamente a la optimización de consultas. Las más importantes son la capacidad de los cursores adaptados para ayudar a decidir si deben mantener un mismo plan para cualquier valor de un parámetro dado y las mejoras en las líneas base de los planes SQL. La primera fue tratada en la sección dedicada a las variables ‘bind’. En cuanto a la segunda, Oracle es ahora capaz de reevaluar un plan si detecta que factores subyacentes como las estadísticas o los parámetros de la base de datos han cambiado y podrían llevarle a trazar un plan inadecuado.

Integrado en Oracle está su optimizador interno, que trata de averiguar para cada consulta SQL cuál es el método más efectivo de ejecución teniendo en cuenta los métodos de acceso a los objetos referenciados y las propias condiciones de la consulta. Para lograr este objetivo, el optimizador repite cinco operaciones en cada proceso de optimización (Oracle Docs):

I. Evaluación de expresiones y condiciones. En primer lugar, el optimizador evlúa las expresiones y condiciones que contengan constantes.

II. Transformación de la sentencia. En sentencias complejas que involucren por ejemplo subconsultas o vistas, el optimizador puede considerar reescribirlas en forma de join equivalente.

III. Elección de la meta del optimizador. Según se persiga mejorar los tiempos de respuesta o el consumo de recursos, el optimizador podrá variar sus decisiones. Podrá tomar el valor 'all_rows', 'first_rows', 'choose' o 'rule'.

IV. Elección de los métodos de acceso o 'access paths'. Para cada tabla referenciada en la sentencia SQL, el optimizador debe escoger un forma de acceso a los datos. Existen seis métodos básicos de acceso: 1) full table scans, 2) rowid scans, 3) index scans, 4) cluster Access, 5) hash Access y 6) sample table scans. El optimizador escogerá cualquiera de ellos basándose en dos factores: 1) la disponibilidad del método de acceso para la sentencia SQL en cuestión, y 2) la estimación del coste de ejecutar dicha sentencia SQL empleando cada uno de los métodos o combinación de ellos (en este segundo factor se incluiría el uso de estadísticas).

V. Elección del método de integración de resultados ('join orders'). Cuando hay involucradas al menos dos tablas de las cuales se extraen resultados que deben integrarse entre sí, el optimizador escoge el orden de esta integración, diferenciando las 'driving tables' como las primeras en comenzar este proceso.

4.2.2 SQL Developer

SQL Developer es la herramienta recomendada por Oracle complementaria a Oracle DB XE 11g R2. Aunque la descarga del software es gratuita, existen funcionalidades que requieren licencia de pago, como el "SQL Advisor", que es el asistente de mejora de consultas. Sobre esta función no se conceden períodos de prueba gratuitos, por lo que no ha sido contemplada en el estudio.

Resulta especialmente útil a la hora de generar 'explain plans'. Al solicitarlo, SQL Developer muestra una tabla cuyas columnas contienen información relevante sobre la ejecución de la consulta, como la operación hecha, si se han aplicado hints, el coste que ha tenido cada operación, y el propietario del objeto entre otros:

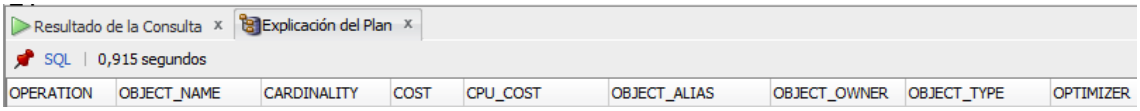


Ilustración 9: SQL Developer - Parámetros del plan

En contraposición, el explain plan generado por consola mediante el comando "SELECT PLAN_TABLE_OUTPUT FROM TABLE(DBMS_XPLAN.DISPLAY());" sólo muestra el valor hash del plan ('plan hash value'), el Id de la operación, el nombre de la operación, el nombre del objeto implicado, el número de filas y el número de bytes extraídos.

SQL Developer es una herramienta oficial de Oracle ideada para facilitar el manejo de las bases de datos, ofreciendo una interfaz gráfica más adaptada al usuario que la consola. La sección de generación de consultas, en las hojas de trabajo propias de cada usuario, se distinguen dos subpartes: 1) Hoja de trabajo, donde se redactaría la consulta en cuestión, y 2) Generador de consultas, que pretende aportar una ayuda adicional al usuario en el proceso:

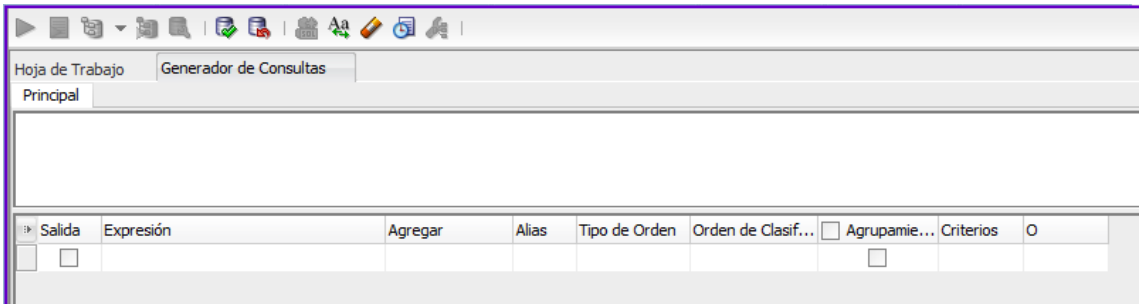


Ilustración 10: SQL Developer - GUI editor y generador de consultas

Como es posible observar en la imagen, el generador de consultas está exclusivamente formado por una tabla que debe ser completada por el usuario. Carece de cualquier ayuda visual sobre las tablas o mensajes emergentes que simplifiquen el proceso de rellenado de la tabla.

SQL Developer ofrece una versión de la función SQL*Plus Autotrace mejorada, que permite detectar en qué puntos de la ejecución se han consumido más recursos: cuántos buffer gets ha usado, cuánto tiempo ha tardado y cuántas lecturas de disco y escrituras han sido necesarias. Para habilitar esta funcionalidad si no está permitida, consultar los anexos. Respecto a la función de SQL*Plus, en SQL Developer es posible comprimir y expandir secciones del plan de ejecución y comparar varios entre sí.

Algunas de las restricciones de esta herramienta y su funcionalidad "Script runner" son las siguientes, extraídas del propio manual de SQL Developer:

- No es posible utilizar variables 'bind' de tipo VARCHAR2, NUMBER ni DATE, que son los tipos de los campos en las tablas más empleadas:

```
SQL> desc jobs;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 JOB_ID                                    NOT NULL VARCHAR2(20)
 JOB_TITLE                                 NOT NULL VARCHAR2(45)
 MIN_SALARY                                         NUMBER(6)
 MAX_SALARY                                         NUMBER(6)

SQL> desc employees;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 EMPLOYEE_ID                               NOT NULL NUMBER(6)
 FIRST_NAME                                         VARCHAR2(20)
 LAST_NAME                                 NOT NULL VARCHAR2(25)
 EMAIL                                     NOT NULL VARCHAR2(25)
 PHONE_NUMBER                                       VARCHAR2(20)
 HIRE_DATE                                 NOT NULL DATE
 JOB_ID                                    NOT NULL VARCHAR2(20)
 SALARY                                             NUMBER(8,2)
 COMMISSION_PCT                                     NUMBER(2,2)
 MANAGER_ID                                         NUMBER(6)
 DEPARTMENT_ID                                      NUMBER(4)

SQL> desc departments;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 DEPARTMENT_ID                             NOT NULL NUMBER(4)
 DEPARTMENT_NAME                           NOT NULL VARCHAR2(30)
 MANAGER_ID                                         NUMBER(6)
 LOCATION_ID                                        NUMBER(5)
```

Ilustración 11: Descripción tabla 'JOBS'

- La función “describe” puede no funcionar con todos los objetos, aunque sí lo haga con SQL*Plus.
- Los comentarios SQL*Plus son ignorados.
- En conclusión, si existe la posibilidad de utilizar SQL*Plus (esto es, consola), es mejor que la función ‘Script Runner’ de SQL Developer.

SQL Developer permite rastrear las sesiones que están teniendo lugar actualmente activando la herramienta “Rastrear sesiones” en el menú “Herramientas”. Una vez dentro, aparecerá como mínimo una sesión, que es la que se tiene abierta mediante SQL Developer (Module = SQL Developer). Si también hay otras sesiones, por ejemplo mediante consola (Module = SQL*Plus), estas también aparecerán registradas:

	INST_ID	SID	SERIAL	SQL_ID	Username	Seconds in Wait	Command	Machine	OS ...	Sta...	Module
1	1	67	33	1uf8n1vwkcyu2	HR	0	SELECT	Algodonales-dv6	Raquel	active	SQL Developer
2	1	114	255	(null)	HR	(null)	(null)	WORKGROUP\ALGODONALES-DV6	Alg...	inactive	SQL*Plus
3	1	136	85	(null)	HR	(null)	(null)	WORKGROUP\ALGODONALES-DV6	Alg...	inactive	SQL*Plus

Ilustración 12: SQL Developer - Registro de sesiones

Desde este mismo menú es posible rastrear otras sesiones e incluso finalizarlas, si se tienen los permisos necesarios. En ambos casos, se despliega una ventana emergente en la que se traduce la potencial orden a SQL:

- Rastrear la sesión: begin dbms_monitor.session_trace_enable(136, 85, TRUE, TRUE); end;
- Finalizar la sesión: ALTER SYSTEM KILL SESSION '114, 255, @1' IMMEDIATE

Para interpretar correctamente esta información es necesario tener en cuenta que la conexión concreta que se esté usando para comprobar el estado de todas las conexiones a la base de datos influye en la información mostrada. Por ejemplo, en la imagen anterior, SQL Developer indica que la sesión activa es la designada por el SID 67, mientras que si se utiliza la conexión vía consola con SID 136 se señalará esta y no la 67 o la 114 como activas.

Ventajas:

- Permite ver rápidamente toda la información relativa a una tabla del esquema
- Permite formatear el código de las sentencias para adaptarlo a otros proveedores. Aunque por defecto es Oracle, también permite adaptarlo a Access, DB2, SQL Server y Sybase.
- En el menú Preferencias > Base de datos > Compilador, puede escogerse el nivel de optimización PL/SQL (0, 1, 2 o 3). Cada cambio en este parámetro añade las siguientes líneas al log:

76	xe_hr	1	alter session set PLSQL_DEBUG=true
75	xe_hr	9	alter session set PLSQL_OPTIMIZE_LEVEL=1
74	xe_hr	6	alter session set statistics_level=TYPICAL

Ilustración 13: SQL Developer - Efectos de cambios en el nivel del optimizador PL/SQL

- Permite configurar qué información mostrar tanto en los explain plan como en los autotrace en la opción de Preferencias > Database: Autotrace/Explain Plan.
- Permite comparar salidas de su función 'Autotrace', donde además se marcan en rojo los valores que no coinciden. Pero sólo permite comparar dos entre sí.
- Los mensajes de error son claros y aportan recomendaciones para solucionarlo en general. En muchos casos los errores se deben a falta de privilegios, e indican qué privilegios en concreto debe tener el usuario para efectuar una determinada tarea.
- Si se cuenta con los privilegios suficientes, es posible gestionar otras sesiones desde esta herramienta, que permitiría rastrearlas o incluso finalizarlas.

Inconvenientes:

- Los objetos mostrados en una conexión dependen del rol con el que fueron creados. Por ejemplo, si al crear una conexión no se especificó ningún rol por defecto, y por consola se crea una tabla usando el rol SYSDBA, esta tabla no será mostrada en SQL Developer.

- Al crear una conexión nueva, no permite que el rol por defecto sea SYSDBA.

- En ocasiones su rendimiento disminuye, las acciones se ejecutan con lentitud y la interfaz gráfica no muestra todos los elementos o entremezcla secciones.

- No permite ver el modelo global en un usuario. Es decir, dentro de una conexión asociada a un usuario, la herramienta permite ver la sección del modelo en la que está incluida una determinada tabla, pero no permite ver a un usuario su propio modelo al completo. De estar disponible, verificar las interrelaciones y los efectos que puede tener un cambio de un objeto del modelo en todo el modelo sería más rápido y seguro. Por ejemplo, en el esquema HR, la sección "Model" de la tabla "LOCATIONS" sólo muestra su relación directa con las tablas "DEPARTMENTS" y "COUNTRIES", cuando existen también "EMPLOYEES", "JOB_HISTORY", "JOBS" y "REGIONS".

- No permite realizar cambios en caliente. Al intentar añadir una nueva fila a una tabla, cuando detecta un error como una violación de la clave primaria, no permite realizar los cambios pertinentes en el momento y continuar la operación, sino que es necesario cancelar la operación y después de subsanar el error repetir la incorporación.

- Los cambios de configuración realizados a través de SQL Developer no se sincronizan con la consola abierta SQL*Plus.

- 'Permite' ejecutar funciones y trabajar con elementos que no están habilitados, es decir, ejecuta sin mostrar error alguno, lo que hace creer al usuario equivocadamente que ha realizado una determinada acción. Esto sucede por ejemplo al tratar de utilizar la caché de resultados en la versión 11g XE.

Por último, se añade al análisis específico de SQL Developer el estudio de usabilidad en interfaces de usuario desarrollado por el danés Jakob Nielsen (Nielsen Norman Group). Para cada una de las diez heurísticas se hace una valoración numérica comprendida entre 0 y 10 según el grado de completitud y una escrita que justifica la anterior. Para más detalles acerca del significado de cada heurística, se recomienda consultar el glosario.

Heurística	*SQL Developer*	
	Valoración	*Comentario*
1 – Visibilidad del estado del sistema.	8	Los aparatos están correctamente nombrados y señalados, el cursor y efectos visuales permiten conocer el estado del sistema. Cada operación realizada es notificada al usuario, y mientras la ejecuta también se indica sombreando la pantalla o con símbolos comúnmente aceptados de espera.
2 – Correspondencia entre el sistema y el mundo real.	7,75	Los iconos son familiates, y en caso de que no lo sean aparecen convenientemente etiquetados. Pueden emplearse atajos de teclado y los colores respetan los estándares esperados (verde, amarillo y rojo). Sin embargo, a veces los mensajes no están adecuados al lenguaje del usuario y no ofrecen la ayuda necesaria para completar una acción.
3 – Libertad y control del usuario.	8,5	Es sencillo de manejar, y el usuario puede alterar el orden de los botones y las pestañas. En caso de aparecer más de una ventana pueden manejarse de forma sencilla. Hay funciones de hacer y deshacer, es posible interrumpir o cancelar operaciones en curso, y los menús son de un tamaño adecuado en general (menos el menú 'ver', que es demasiado extenso). En los cuadros de diálogo que incluyan varios pasos, sí es posible retroceder.
4 – Consistencia y estándares.	8,5	Los iconos están etiquetados y aunque hay variedad, no confunden al usuario. Se permite scroll vertical y horizontal para visualizar la información. Se respetan los estándares de colores, y las opciones están agrupadas por funcionalidades y ordenadas alfabéticamente dentro de cada subgrupo.
5 – Ayuda a los usuarios para reconocer, diagnosticar y recuperarse de errores.	6	Los mensajes de error pueden ser demasiado genéricos, por lo que el usuario necesita consultar otras fuentes externas de información para llegar a comprender los errores y cómo solucionarlos. El tono y lenguajes empleados son adecuados. Los niveles de experiencia del usuario no influyen en la interacción con el sistema, por lo que los mensajes y las operaciones tienen el mismo nivel de dificultad para todo tipo de usuarios.
6 – Prevención de errores.	7	No se permite al usuario aplicar acciones que están desactivadas, antes de ejecutar

		una acción que puede poner en peligro el sistema se solicita confirmación al usuario.
7 – Reconocimiento antes que recuerdo.	8,25	El significado de los objetos es sencillo de recordar, están organizados de forma lógica, se distinguen claramente las operaciones que puede ejecutarse de las que no, los parámetros que se pueden modificar y las casillas que son opcionales marcar. Todo ello organizado en espacios suficientes que permiten distinguir claramente los elementos. Menús sencillos de navegación.
8 – Flexibilidad y diseño minimalista.	7,5	Sí se pueden crear sinónimos propios, establecer una configuración personalizada. No soporta varios tipos de usuario según su grado de conocimiento.
9 – Estética y diseño minimalista.	8	Mismos colores aplicados consistentemente en toda la aplicación, elementos separados entre sí, interfaz no completa de elementos, por lo que no agobia al usuario, títulos de longitud adecuada y representativos.
10 – Ayuda y documentación.	6	Requieren mejoras. Aunque está disponible en cualquier vista, los contenidos a veces están incompletos o no detallan lo suficiente.
Total	7,55	

Tabla 7: Valoración heurística de usabilidad SQL Developer - Lista Pierrotti

4.2.4 TOAD

Toad es la herramienta frente a la cual se compara SQL Developer en este estudio. Toad ofrece varios productos aplicables a Oracle, que son: Base Edition, Professional, Xpert Edition, Toad Development Suite for Oracle, Toad DBA Suite for Oracle (also Exadata Edition and RAC Edition). Se ha seleccionado "Toad for Oracle Xpert Edition", usando una licencia temporal de 30 días.

La edición experto de Toad incluye las característica del producto inmediatamente anterior (edición profesional) e incorpora un SQL integrado sintonizado con un optimizador SQL, que es la rama de funciones relevante para este estudio. Para ejecutar correctamente Toad en el equipo, algunos privilegios son necesarios según qué edición se esté empleando. La versión 12.9 con la que se realiza el estudio sólo requiere permisos especiales para las ediciones básica y DB suite, por lo que no son relevantes en este caso.

Una vez creada una consulta, el panel de opciones desplegado con el botón derecho permite configurar el escritorio de trabajo e indicar en qué posición de la pantalla es preferible que aparezcan los resultados. Esta libertad de configuración es extensible a los resultados, en los que puede seleccionarse qué columnas mostrar y

ocultar, el grado de detalle (es posible mostrar incluso porcentajes) y el orden, pero no es posible aplicar cambios para que sean mostrados de forma más amigable. Es decir, los códigos de colores y rayados en el explain plan que aparecían en SQL Developer son sustituidos por blancos y negros en Toad, lo que hace difícil la lectura de algunas líneas.

Para ejecutar una sentencia, Toad ofrece cinco opciones al desplegar el menú del botón derecho:

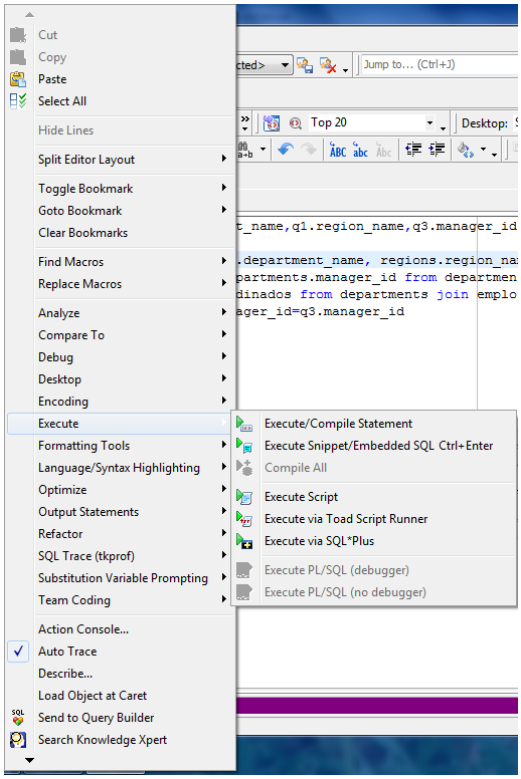


Ilustración 14: TOAD - Opciones desplegadas

En lo relativo a la optimización de consultas, que era el principal reclamo de esta herramienta, figuran tres alternativas:

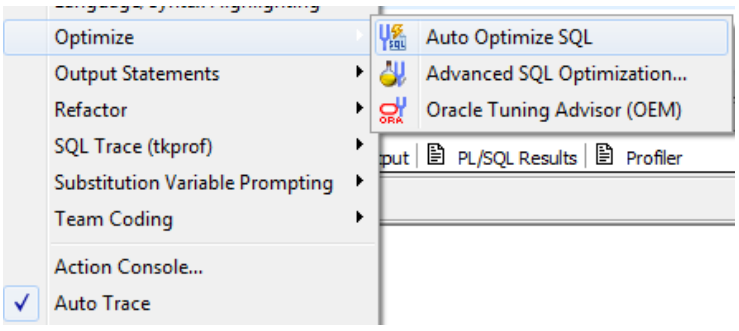


Ilustración 15: TOAD - Opciones de optimización

- Auto Optimize SQL [AO SQL]: es la función más rápida, sencilla e intuitiva de usar en Toad al optimizar una sentencia SQL. El usuario sólo debe indicar si desea imponer un tiempo límite de generación de alternativas (como forma de

asegurar que esta búsqueda de alternativas finalizará en un tiempo conocido) y si la sentencia debe ser refactorizada antes de su análisis. Si bien el administrador puede influir poco en el análisis, esto se compensa con una interfaz clara dividida en tres secciones: lista de alternativas ordenadas en base a un elemento, sentencia de partida y sentencia alternativa con la que se está comparando. Por ejemplo:

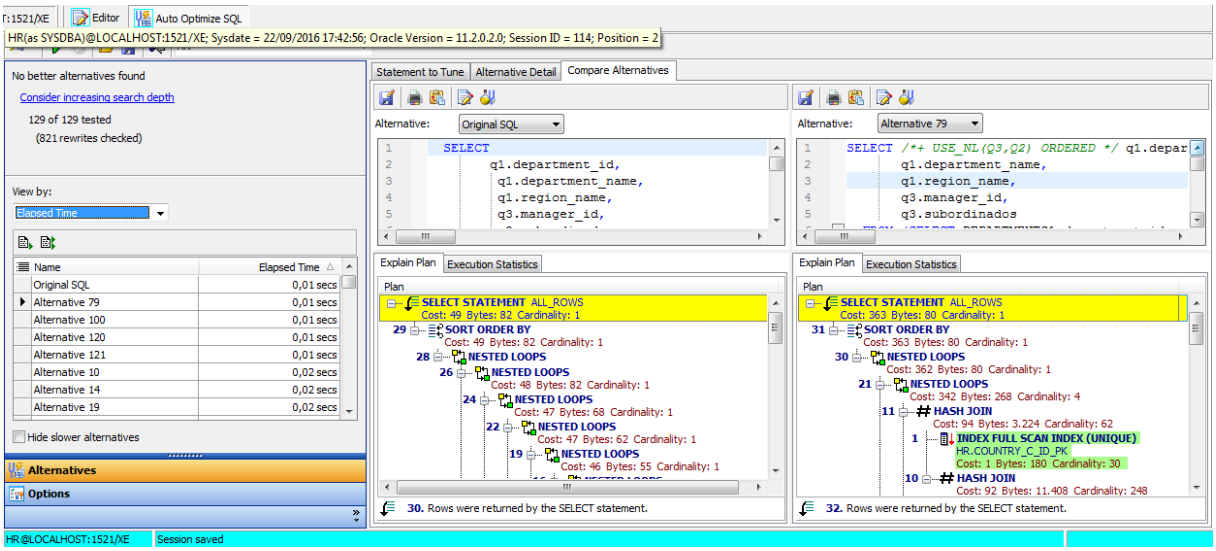


Ilustración 16: TOAD - Vista de la función Auto Optimize SQL

- Advanced SQL Optimization [Adv SQL Opt]: esta opción abre un módulo complementario a Toad llamado Dell SQL Optimizer for Oracle, en el que el usuario adquiere el control sobre la configuración del optimizador. Puede indicar el grado de variabilidad del entorno en que se ejecuta la sentencia, así como el nivel de 'inteligencia' del optimizador y la generación de índices. A diferencia de la anterior funcionalidad, los datos de la ejecución de cada alternativa son mostrados simultáneamente en una tabla comparativa, lo que hace este análisis más completo (en el anterior se corre el riesgo de obtener resultados pésimos en otros campos a costa de mejorar uno en concreto, lo que llevaría a una toma de decisiones desaconsejable). Imágenes ilustrativas de esta función han sido incluidas en los anexos 9.3 [9.3 Alternativas de las consultas TOAD].

- Oracle Tuning Advisor (OEM - Oracle Enterprise Manager): El OEM original de Oracle permite gestionar la nube, aplicaciones y bases de datos, pero no es aplicable en entornos que requieran gestionar simultáneamente un gran número de bases de datos. En este aspecto interviene TOAD, que completa la funcionalidad añadiendo gestión del rendimiento, mantenimiento de bases de datos y gestión de cambios. Requiere licencia independiente de Toad para poder ser utilizada, por lo que no se incluye en este estudio.

La optimización mediante la función Advanced SQL Optimizer despliega en primer lugar una ventana [Ilustración 57: Adv SQL Opt (I)] en la que el usuario debe escoger el modo del optimizador: 'SQL Rewrite', 'Plan Control' y 'Batch Optimize SQL', de los cuales para este estudio sólo es relevante el primero.

La reescritura SQL se basa en el motor de optimización 'SQL Optimizer's Artificial Intelligence'. En primer lugar el optimizador analiza la sentencia SQL original y a continuación genera una lista de sentencias semánticamente equivalentes. Sobre esta lista se aplican una serie de hints de Oracle (que pueden delimitarse, si no se quieren emplear todos) para extender al máximo esta lista de alternativas.

El modo 'Plan Control' está enfocado para ser empleado en situaciones en las que no se dispone del código fuente, y por lo tanto todas las posibles mejoras se generan sobre el plan de ejecución (sin llegar a cambiar el código). Para ello utiliza la funcionalidad 'SQL Plan Management' introducida en Oracle 11g. El proceso de generación de planes alternativos es similar al del método de reescritura anterior.

Por su parte, la optimización batch se emplea para optimizar un conjunto de trabajos donde cada uno de ellos contiene diversas sentencias SQL. El optimizador analizará todos los 'jobs', seleccionará aquellas sentencias de todas las incluidas que más ineficientes sean y tratará de optimizarlas.

Entre las opciones de configuración figuran los niveles de 'inteligencia' del optimizador y la generación de índices, llamados 'Optimizer Intelligence Level' e 'Index Generation Intelligence Level'. Estos valores determinan cómo de exaustiva será la búsqueda de nuevos escenarios y qué hints o características debe tener en cuenta a la hora de generarlos.

- 'Optimizer Intelligence Level': el valor de este parámetro afectará a los valores contenidos en las opciones de optimización, hints y cuotas ('quota'). Algunos ejemplos de estas opciones son:
    - Opciones de Optimización: ignorar las vistas del esquema SYS, transformación de las consultas en vistas inline, permitir reescritua utilizando la sintaxis ANSI-92 o manteniendo la original, etc.
    - Opciones de Hints: se incluye una lista de hints que estarán disponibles para generar alternativas. Cuanto más bajo es el nivel de inteligencia, más control tiene el usuario para elegir qué hints quiere que sean tenidos en cuenta y cuáles no. Sin embargo, a mayor nivel, hay más hints que obligatoriamente serán empleados.
    - Opciones de Cuotas: número máximo de sentencias SQL que se pueden generar aplicando las reglas de transformación de sintaxis por el motor de inteligencia artifical del optimizador, el número máximo de sentencias SQL que pueden ser generadas por la aplicación de hints de Oracle para

la optimización, y el número máximo de reordenaciones de joins durante la optimización.

- 'Index Generation Intelligence Level': su valor afecta a los valores de las opciones de la generación de índices, entre las que se encuentran el tamaño de bloque utilizado en los muestreos (sampling), el número máximo de datos a utilizar en el muestreo, y el número máximo de alternativas de índices que se generarán.

Tras seleccionar el método de optimización, en el diálogo dedicado a los síntomas [Ilustración 61: Adv SQL Opt (III)], es posible definir con más detalles el entorno en el que se desplegará la consulta: con qué fin de utiliza la consulta a optimizar, cómo se usa (estática o dinámicamente), la frecuencia de uso y síntomas de bajo rendimiento (consumo de CPU, IO - Entradas y Salidas o tiempo excesivo). Esto servirá al optimizador para adaptar la búsqueda a dichas necesidades. En este caso, al tratarse de un entorno de pruebas, se ha optado por mantener los valores por defecto.

TOAD requiere privilegios Oracle específicos para ejecutar algunas de las funciones en el optimizador SQL. A continuación se presenta en forma de tabla los módulos, la funcionalidad y los respectivos privilegios relacionados con la optimización de consultas, y en los anexos se ha incluido el script propuesto por TOAD para su concesión [9.5 Script TOAD - Privilegios sobre la base de datos]:

Module	*Functionality*	*Privilege*
All Modules	Trace setup options: Enable collection of Oracle trace statistics	Requires ALTER SESSION privileges. Requires access to the following views: SYS.V_$SESSION SYS.V_$PROCESS
All Modules	Retrieve DBMS_XPLAN	Requires access to the SYS.DBMS_XPLAN package.
All Modules	General	If the Oracle init parameter O7_DICTIONARY_ACCESSIBILITY for Oracle 8 or later is set to false, you cannot access objects under SYS even if you have SELECT ANY TABLE privileges. In this case, you need SELECT ANY DICTIONARY privileges or SELECT_CATALOG_ROLE to access the objects under SYS.
Optimize SQL (SQL Rewrite)	Alter session parameters for executing SQL	Requires access to SYS.V_$PARAMETER view.
	Generate virtual indexes	Requires Oracle 8i or later.
	Check existing translation for your SQL	Requires Oracle 12c or later. Requires access to the SYS.ALL_SQL_TRANSLATIONS view.
	Create SQL Translation Profile	Requires Oracle 12c or later. Requires CREATE SQL TRANSLATION PROFILE privileges.
	Deploy outlines	Requires Oracle 8i or later. Requires CREATE ANY OUTLINE and DROP ANY OUTLINE privileges. Requires access and UPDATE privileges to the following views: OUTLN.OL$HINTS

		OUTLN.OL$ OUTLN.OL$NODES
	Register SQL Translation	Requires Oracle 12c or later. Requires access to the SYS.DBMS_SQL_TRANSLATOR package. Requires access to the SYS.ALL_SQL_TRANSLATION_PROFILES view.
Optimize SQL (Plan Control)	Open database connection	Requires Oracle 11g or later.
	Retrieve execution plans, generate alternative plans	Requires ADMINISTER SQL MANAGEMENT OBJECT privileges. Requires access to the following packages: SYS.DBMS_SQL SYS.DBMS_SPM SYS.DBMS_XPLAN Requires access to the following views: SYS.DBA_SQL_PLAN_BASELINES SYS.V_$SQLAREA SYS.V_$SQLTEXT_WITH_ NEWLINES
Optimize SQL and Batch Optimize SQL	Execution method option: Run on server setting	Requires access to the SYS.DBMS_SQL package.
	Retrieve run-time statistics	Requires access to the following views: SYS.V_$MYSTAT SYS.V_$STATNAME SYS.V_$PARAMETER
	Retrieve actual plan	Requires ALTER SESSION privileges. Requires access to the SYS.DBMS_XPLAN package. Requires access to the following views: SYS.V_$SQLAREA SYS.V_$ SQL_PLAN_STATISTICS_ALL SYS.V_$SESSION
	Capture bind values from database	Requires Oracle 10g or later. Requires access to the following views: SYS.V_$SQLAREA SYS.V_$SQL_BIND_CAPTURE
Optimize Indexes	Recommend indexes	Requires Oracle 8i or later. Requires access to the SYS.V_$SESSION view.
	Access the AWR	Requires Oracle 10g or later. Requires access to the following system views: SYS.DBA_HIST_SNAPSHOT SYS.DBA_HIST_SQLTEXT SYS.DBA_HIST_SQLSTAT
	Display the control information for the Workload Repository	Requires access to the following system view: SYS. DBA_HIST_WR_CONTROL
	Display the SQL summary for the Workload Repository	Requires access to the following system view: SYS. DBA_HIST_SQL_SUMMARY
	Access Foglight PA Repository	Requires access to the following tables: QUEST_SC_ACTION_DIM QUEST_SC_CLIENT_INFO_DIM QUEST_SC_MODULE_DIM QUEST_SC_SQL_STAT_FACT QUEST_SC_SQL_SYNTAX_DIM QUEST_CTRL_PYRAMID_LEVELS QUEST_DB_USER_DIM

		QUEST_INSTANCE_DIM QUEST_PROGRAM_DIM QUEST_TIME_DIM
	Access the SGA	Requires access to the SYS.V_$SQLAREA view.

Tabla 8: Privilegios en TOAD relacionados con la optimización. Fuente: Manual oficial de Toad integrado

Además de las ya mencionadas ventajas aportadas por TOAD, a lo largo de las pruebas realizadas con esta herramienta se han descubierto otras que son especialmente atractivas en procesos de análisis de comportamiento de consultas:

- El navegador de tracefile mejora en gran medida a TKPROF. Muestra estadísticas de todas las ejecuciones, permite ordenar los datos de estas estadísticas por columnas, y presenta los resultados agrupados por áreas de interés como tipos de esperas y operaciones realizadas sobre ellos.

- Si se activa la funcionalidad tracefile - tkprof, es posible rastrear el origen de los resultados mostrados, ofreciendo un conocimiento más completo de los mecanismos de monitorización [Opción 'display file contents where data of selected row was found' del botón derecho].

- Permite seleccionar qué campos del tracefile se desea que aparezcan y cuáles no [Opción 'select visible columns' del botón derecho].

- Al pasar el cursor por encima de un campo del navegador tracefile aparece un mensaje en el que se describe qué representa dicho parámetro. Es especialmente útil cuando el nombre de la columna no es el nombre completo del parámetro.

- En la función Auto Optimize SQL, la comparativa de estadísticas sigue un código de colores (verde cuando es favorable y rojo cuando no) e incluye en qué porcentaje mejora o empeora a la ejecución con la que se compara:

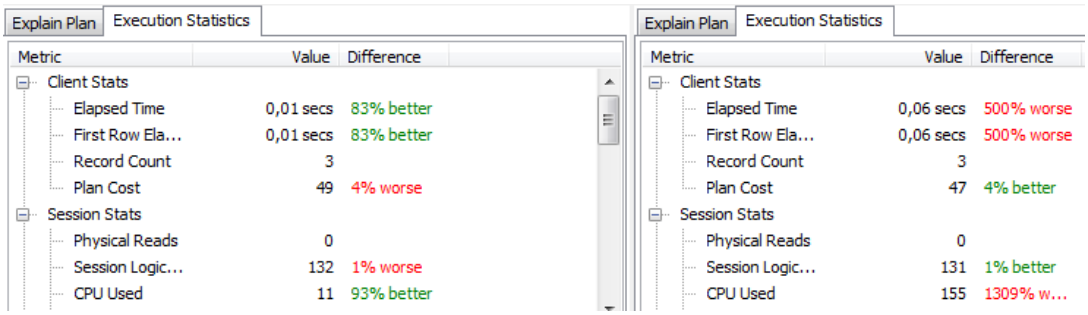


Ilustración 17: TOAD - Execution statistics en Auto Optimize SQL

- El usuario puede elegir el formato en que desea ver un plan de ejecución y qué granularidad en los resultados en el formato del dbms_xplan. El tercer modo, 'plain english', es especialmente útil para interpretar correctamente el plan, porque explica cada operación realizada:

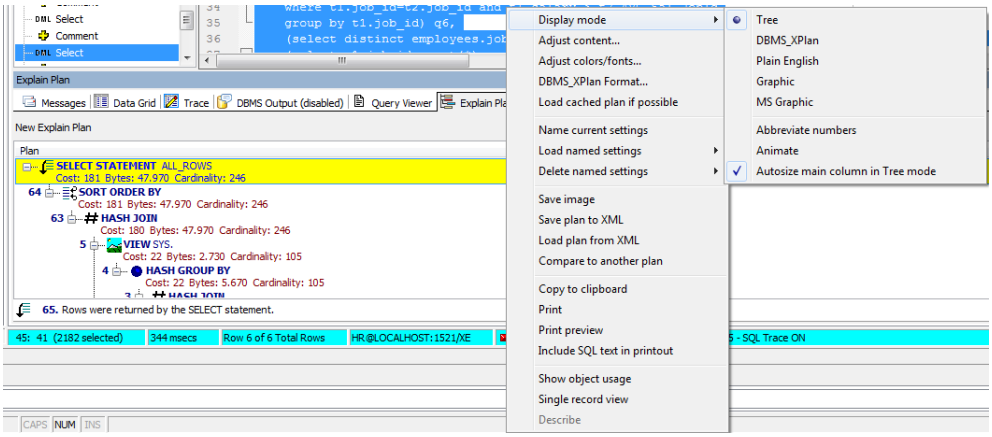


Ilustración 18: TOAD - Modos de presentación del explain plan

- 'Tree Plan' — Muestra el plan en forma de árbol junto a los pasos enumerados.
- 'DBMS_XPlan (Formatted)' — Muestra el plan usando la salida de la función dbms_xplan, al que le añade su propio formato.
- 'DBMS_XPlan (Plain Text)' — Muestra el plan usaodn también la función dbms_xplan, pero esta opción lo muestra en texto claro (con frases).
- 'Graphic Plan y MS Graphic Plan' — Son dos representaciones gráficas del plan que incluyen también la numeración de pasos.
- 'Plain Language Plan' — En una lista numerada describe los pasos que se han llevado a cabo, en lenguaje natural.

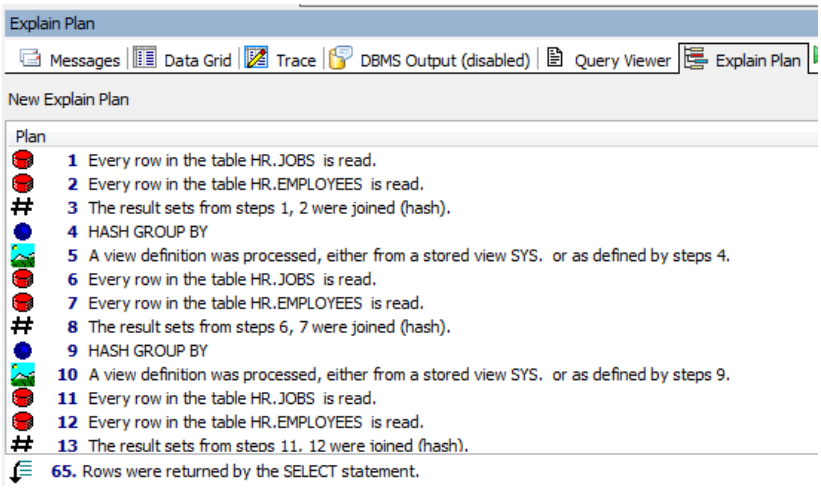


Ilustración 19: TOAD - Display Mode = Plan English en Explain Plan

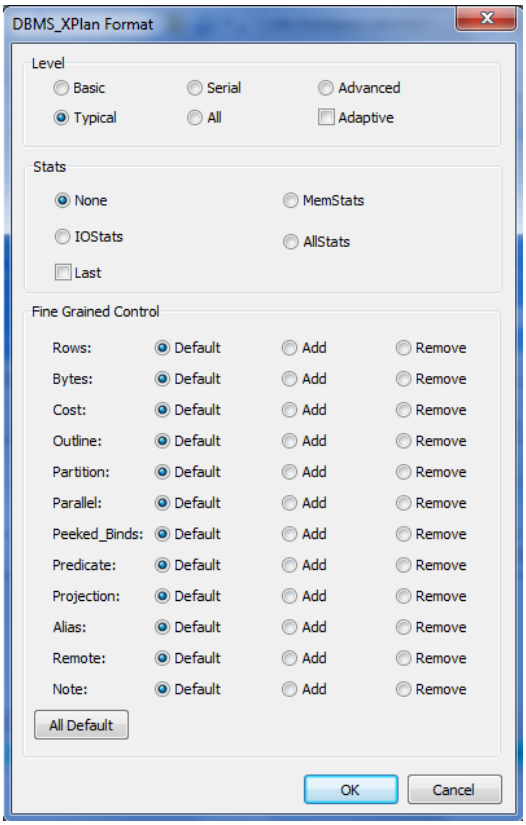


Ilustración 20: TOAD - Configuración por defecto del formato dbms_xplan

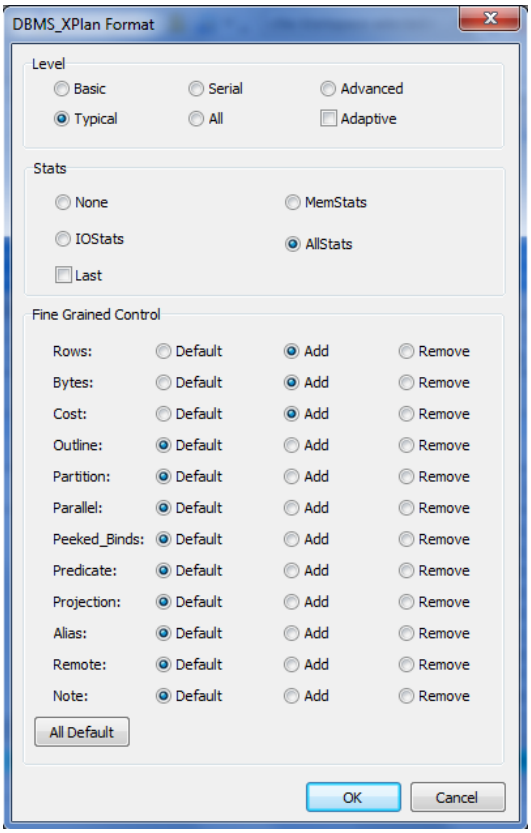


Ilustración 21: TOAD - Configuración personalizada del formato de dbms_xplan

A raíz de las pruebas realizadas en TOAD 12.9, se han detectado varias limitaciones que dificultan su uso:

- En ocasiones los errores no son lo suficientemente detallados y por lo tanto son poco útiles al tratar de subsanarlo. Dos ejemplos son 'table doesn't exist' y 'ORA-00905: missing keyword', que no indican qué tabla de las referenciadas no existe o qué palabra concreta falta.

- Los mensajes de error se quedan a veces 'atascados' y aparecen al ejecutar una sentencia diferente, haciendo incluso referencia a líneas de error que no existen en la consulta porque es más corta.

- Una vez añadida una conexión, no es posible eliminarla, sólo modificarla. Con ello se corre el riesgo de mantener fallos de conexión que podrían eliminarse al retirar una conexión de la lista.

- En la opción Advanced SQL Optimization, durante la generación de alternativas teóricamente es posible pausar el proceso y retomarlo en el mismo punto. En la realidad, no hay ningún botón visible que realice esta acción.

- Sólo en las estadísticas de la función Auto Optimize SQL se reflejan los resultados en formato número - unidad. Un mismo parámetro, por ejemplo 'elapsed time', aparece en AO SQL como "0'01 secs" y cambia su formato a "0:00:00'01" en Adv SQL Opt.

- Los parámetros de rendimiento mostrados en las estadísticas de los planes de ejecución varían según se observe desde AO SQL o Adv SQL Opt. Por ejemplo, el modo del optimizador sólo es visible en el primero.

- En la función Advance SQL Optimizer no indica visiblemente cuánto tiempo ha transcurrido en la búsqueda de nuevos escenarios, sino que es el usuario quien debe navegar por las opciones y hacer scroll hasta encontrar el dato, lo que reduce la usabilidad de la herramienta. En Auto Optimize SQL sí que lo indica mientras lo ejecuta, pero no al final, por lo que el usuario debe prestar atención constante al proceso.

- No permite la ejecución en paralelo de más de un proceso de búsqueda de mejores alternativas, por lo que la ejecución de las pruebas es secuencial.

- Se pueden establecer restricciones de tiempo e incluso de número de alternativas que se permiten generar como máximo al explorar nuevos escenarios, pero no permite establecer restricciones respecto a otros parámetros, como el uso de CPU, el coste del plan o el número de operaciones de lectura empleadas.

- Durante el proceso de exploración de nuevos escenarios, que puede extenderse durante horas, no es posible guardar los escenarios ya hallados periódicamente, lo que expone al sistema a un gran riesgo, ya que si se da un corte de servicio, todo el trabajo realizado hasta el momento se perdería.

Al igual que en la anterior herramienta, se completa el análisis de TOAD empleando las diez heurísticas de usabilidad propuestas por Jakob Nielsen para el diseño de interfaces gráficas de usuario. La valoración cuantitativa está basada en la lista Pierrotti (Pierrotti Nielsen Heuristics), que especifica una lista de aspectos por cada heurística cuya presencia o no ayuda a evaluar en qué grado se cumple dicha heurística (consultar Webgrafía):

Heurística	*TOAD*	
	Valoración	*Comentario*
1 – Visibilidad del estado del sistema.	8,8	El usuario es capaz de ver e interpretar correctamente el estado del sistema en la mayoría de los casos. Sin embargo, los errores no permiten ver qué parte concreta es la problemática, los tiempos para de algunas tareas son más elevados de lo que cabría esperar.
2 – Correspondencia entre el sistema y el mundo real.	8,8	Los iconos contienen formas que se corresponden con las funciones y el lenguaje es comprensible por el usuario con frecuencia. El código de colores es el esperado y comúnmente aceptado. Los comandos pueden ser introducidos de abreviada o completa.
3 – Libertad y control del usuario.	8	El usuario es capaz de reordenar las ventanas o paneles y cambiar de uno a otro con facilidad. Ante operaciones de gran magnitud, el usuario debe confirmarlas antes ser aplicadas por el sistema. Aunque hay una opción 'deshacer', no permite deshacer un grupo de operaciones, y tiene un máximo. Sí se permiten cancelar operaciones en curso. También hay atajos de combinaciones de teclas a disposición del usuario. Se puede navegar libremente entre los menús, y se permite establecer una configuración personalizada por defecto.
4 – Consistencia y estándares.	8,3	Se mantiene un formato consistente durante toda la aplicación, aunque se abusa del uso de mayúsculas en operaciones de reformateo. El panel en uso aparece claramente señalado. Sin embargo, aunque los menús sean desplegados verticalmente, en ocasiones son excesivamente largos y sus elementos no están numerados. Por convención del sector, los menús se colocan adecuadamente en el triángulo superior izquierdo, y sus elementos están ordenados alfabéticamente. Existen múltiples tamaños, fuentes y colores a elegir (en los ajustes). Las técnicas de atracción de atención se utilizan puntualmente, en

		las secciones relevantes y sin exceso. Los mensajes y menús aparecen en los mismos lugares aunque se traten de páginas o ventanas distintas, aportando consistencia.
5 – Ayuda a los usuarios para reconocer, diagnosticar y recuperarse de errores.	5,2	Los mensajes de error no siempre ayudan al usuario a solucionarlo, a veces son meros notificadores, y en ocasiones son excesivamente ambiguos y breves. El tono es respetuoso, gramaticalmente correcto, evita palabras malsonantes y no culpabiliza al usuario. Por otro lado, no está preparado para adaptar el nivel de detalle al nivel del usuario (experto vs principiante) y no siempre se incluyen ayudas semánticas o sintácticas, ni la causa del problema ni el grado de gravedad del error.
6 – Prevención de errores.	7,7	La longitud de los campos debe conocerse de antemano, antes de insertar los datos, porque esta información no es visible en la sección de datos. Los elementos de los menús son mutuamente exclusivos entre sí y están incluidos en menús con los que se relacionan coherentemente. Existen valores por defecto en las ventanas de diálogo que son necesarios y se ayuda al usuario a minimizar los errores cometidos mediante funciones como autocompletar código.
7 – Reconocimiento antes que recuerdo.	8,6	Los elementos aparecen agrupados de forma lógica, separados entre sí debidamente y posicionados donde se espera verlos. Sin embargo hay listas demasiado largas y en ocasiones no es sencillo diferenciar los campos obligatorios de los opcionales. El código de colores es consistente en toda la aplicación. Es visible qué elementos son elegibles y cuáles no. En ciertos menús, hay opciones marcadas por defecto, y los paneles complementarios sólo se muestran bajo solicitud.
8 – Flexibilidad y diseño minimalista.	5,5	TOAD no distingue entre usuarios novatos y expertos, por lo que depende del usuario saber utilizar la herramienta y maximizar su uso.
9 – Estética y diseño minimalista.	10	La información relevante para la toma de decisiones aparece en la misma pantalla, los elementos están claramente diferenciados y agrupados, y los títulos son cortos pero representativos.
10 – Ayuda y documentación.	6,3	Los paneles de ayuda son sencillos de identificar, navegar por ellos y luego volver a la tarea, pero la información no siempre es detallada, útil o accesible. Algunos ítems y opciones dentro de los menús son ambiguos y no hay explicación disponible. Depende de la experiencia del propio usuario.
Total	7,7	

Tabla 9: Valoración heurística de usabilidad TOAD - Lista Pierrotti

5. Optimización de consultas en SQL Developer y TOAD: Casos de uso

Esta quinta sección está dedicada a los aspectos prácticos del estudio. Los anteriores apartados tenían como fin introducir la materia cada vez con más detalle, desde los aspectos más generales de la concepción relacional y no relacional de las bases de datos hasta el análisis concreto de SQL Developer y TOAD.

Una vez presentada esta información, se completa el estudio con una serie pruebas basadas en tres casos de uso que servirán para valorar estas herramientas según esta experiencia particular, y no exclusivamente por datos teóricos.

- Las subsecciones del apartado 5.1 están dedicadas a explicar el entorno de pruebas empleado, con el fin de que pueda ser reproducido en el futuro por otros interesados, y los casos de uso de partida. Estos últimos son tres consultas SQL ideadas para ser ejecutadas tanto en SQL Developer y Toad y mejoradas utilizando las funcionalidades que cada una de estas herramientas ofrezca. Aparecen como enunciado y a continuación en SQL.

- Los análisis concretos en SQL Developer y Toad han sido incluidos respectivamente en los puntos 5.2 y 5.3. La descripción de las herramientas ha sido incluida anteriormente en 4.2 Herramientas, por lo que estas se centrarán en la visión técnica de las mismas.

5.1 Diseño de los casos de uso

5.1.1 Objetivos

El objetivo de los casos de uso es idear, diseñar y crear hipotéticas entradas para la aplicación con el fin de evaluar su rendimiento y comportamiento antes de implantarlo en un entorno real. En el caso de este estudio, no sólo se valora individualmente una aplicación, sino dos, de manera que aquella que mejor se ajuste a las necesidades de los estudiantes sea aplicada en su plan de estudios.

Los casos de uso empleados aportan distintos niveles de complejidad, obtenidos mediante la utilización de estructuras más complejas como vistas y subconsultas combinadas con operaciones sobre los datos y distintas cargas de datos solicitadas para recuperar.

5.1.2 Entorno de pruebas

Todas las pruebas han sido llevadas a cabo dadas las siguientes condiciones, que determinan el entorno de pruebas, y que garantizan la validez de los resultados para ser comparados entre ellos:

- El mismo equipo hardware ha ejecutado la totalidad de las pruebas, para asegurar que la mejora no se debía a mejoras en los recursos del sistema o distintas arquitecturas.
- Las herramientas empleadas han sido instaladas independientemente para garantizar el aislamiento de las pruebas de variables externas.
- Se ha aplicado el mismo procedimiento en los tres casos de uso según la herramienta analizada.
- La base de datos sobre la que se ha realizado la batería de pruebas ha permanecido intacta desde antes del comienzo de las pruebas, por lo que no han tenido lugar operaciones de ningún tipo que pudieran afectar a los datos almacenados en ella.

La preparación del entorno en Toad es más compleja debido a la libertad que se concede al usuario para adaptar el escritorio y el entorno de desarrollo a sus necesidades. A continuación se presentan las modificaciones realizadas antes de proceder ejecutar las pruebas en Toad:

- Para permitir ejecutar la función tracefile + TKPROF, es necesario asociar un directorio. En caso de que se rechace realizar dicha operación en el momento actual, Toad muestra el comando que deberá ejecutarse para habilitarlo: `Create or replace directory TOAD_TRACEFILE_DIR as 'C:\ORACLEXE\APP\ORACLE\diag\rdbms\xe\xe\trace'`. Importante recordar que para ello se necesitan privilegios especiales, por lo que antes es necesario conectarse con rol SYSDBA.

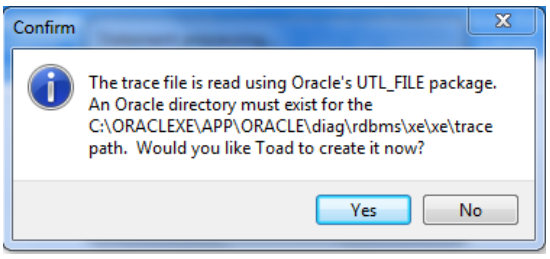


Ilustración 22: TOAD - Error tracefile

- Se activa la funcionalidad tracefile+tkprof, en concreto la que muestre el archivo asociado tras la ejecución:

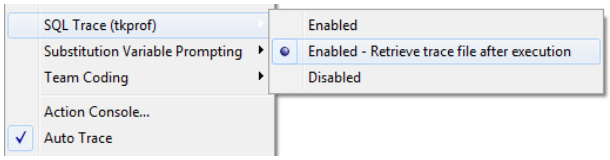


Ilustración 23: TOAD - Activación Tracefile + TKPROF

- De los cinco tipos de ejecución posibles, se utilizará el primero: "Execute/Compile Statement".

- PLS-00201: identifier 'UTL_FILE' must be declared → utilizar el rol SYSDBA en HR para ejecutar las consultas y los análisis de optimización.

- Para evitar el error 'ORA-00942: table or view does not exist' (connect hr as sysdba), según el cual las tablas no existen, aunque realmente estén incluidas en el esquema del usuario HR con el que se establece la conexión se marca la opción "Autoload Tables Based On Object".

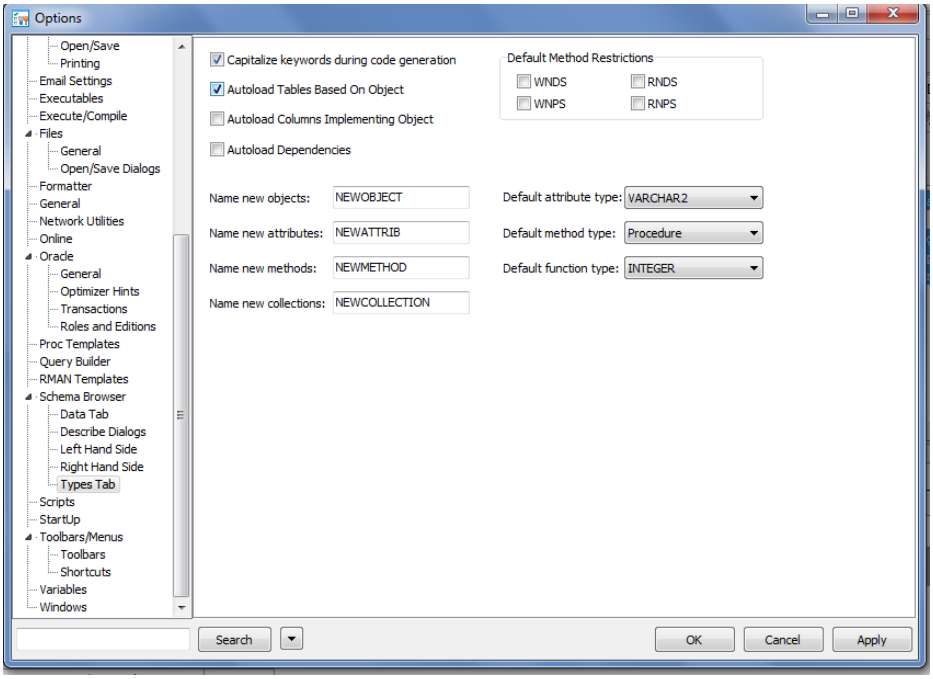


Ilustración 24: TOAD - Opciones de configuración

- La 'plan_table' especificada por defecto en los ajustes de Toad ('toad_plan_table') no existe, de forma que hasta que no se indique una tabla válida (ya sea la ya existente 'plan_table' o una nueva que se cree) no será posible realizar operaciones que requieran el plan de ejecución (como la optimización de consultas). Se selecciona 'plan_table' del esquema SYS como la tabla a usar:

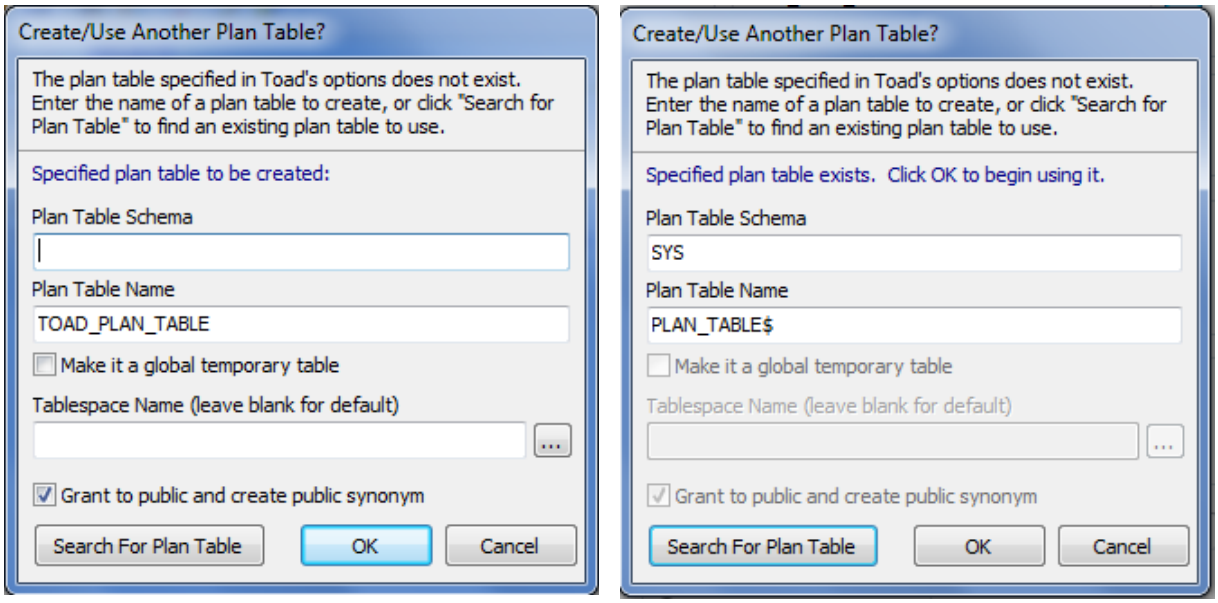


Ilustración 25: TOAD - Definición de Plan Table válida

5.1.3 Elección del esquema BDR y carga de datos

Oracle Database XE 11g r2 incluye en su paquete de instalación un esquema por defecto oculto llamado 'HR', recursos humanos. Se ha escogido este esquema ya creado al poseer la complejidad y la estructura válidas para el desarrollo del estudio. Después de desbloquear la cuenta utilizando permisos de administrador se accede a una base de datos que contiene información relevante para un hipotético departamento de recursos humanos de una empresa. La información aparece recogida en siete tablas: Regiones ('*Regions*'), Países ('*Countries*'), Localizaciones ('*Locations*'), Departamentos ('*Departments*'), Trabajos ('*Jobs*'), Historial de trabajo ('*Job_Hist*') y Empleados ('*Employees*').

Sobre los 107 empleados iniciales y datos insuficientes en otras tablas, se ha ampliado la base de datos de partida, ascendiendo a 5000 empleados, países de las tres grandes regiones (EMEA, Asia y América). Se han aumentado también el número de departamentos, para expandirlos globalmente, e incrementar la variedad de datos.

A pesar de que el esquema HR ya incluye datos, se han añadido más filas con el fin de alcanzar un volumen de procesamiento más elevado, que refleje con más claridad las mejoras en la optimización de consultas.

Para ello se han tenido en cuenta las restricciones inherentes del modelo y las relaciones entre las distintas tablas, especialmente las claves ajenas, como se ve reflejado en la siguiente imagen:

COUNTRIES	DEPARTMENTS	EMPLOYEES	JOB_HISTORY	JOBS	LOCATIONS	REGIONS
COUNTRY_ID	DEPARTMENT_ID	EMPLOYEE_ID	EMPLOYEE_ID	JOB_ID	LOCATION_ID	REGION_ID
COUNTRY_NAME	DEPARTMENT_NAME	FIRST_NAME	START_DATE	JOB_TITLE	STREET_ADDRESS	REGION_NAME
REGION_ID	MANAGER_ID	LAST_NAME	END_DATE	MIN_SALARY	POSTAL_CODE	
	LOCATION_ID	EMAIL	JOB_ID	MAX_SALARY	CITY	
		PHONE_NUMBER	DEPARTMENT_ID		STATE_PROVINCE	
		HIRE_DATE			COUNTRY_ID	
		JOB_ID				
		SALARY				
		COMMISSION_PCT				
		MANAGER_ID				
		DEPARTMENT_ID				

Tabla 10: Interrelación de atributos entre las tablas HR

En este punto se ha utilizado la herramienta web "http://www.generatedata.com/", un generador aleatorio de datos gratuito que además no requiere licencia. Plan de acción:

#1 REGIONS: no requiere modificaciones.

#2 COUNTRIES: para los nuevos países añadidos, tener en cuenta el REGION_ID. A su vez, el COUNTRY_ID será empleado por la tabla LOCATIONS.

#3 LOCATIONS: utiliza los COUNTRY_ID.

#4 JOBS: será utilizado por EMPLOYEES. Primero se obtiene la relación inicial entre departamentos y puestos de trabajo, para ampliar el número de puestos de trabajo y asociarlos a su departamento regional. La relación inicial se obtiene mediante la consulta "select distinct department_id, job_id from EMPLOYEES where (DEPARTMENT_ID is not null) order by DEPARTMENT_ID asc;":

En los datos iniciales (19) no constan puestos de trabajo para cada departamento, por lo que antes de crear nuevos puestos regionales hay que completar los puestos de los departamentos iniciales (es decir, del departamento 120 al 270). Luego hay que crear los puestos de trabajo que también existirán en las otras dos regiones (EMEA incluye Europa - 1 - y Oriente Medio y África - 4). La relación de salarios asumida en la generación de datos es: EMEA = 1'1*América; Asia = 0'75*América.

#5 JOB_HISTORY: es manejada por el disparador "UPDATE_JOB_HISTORY" y el procedimiento "ADD_JOB_HISTORY" cuando se actualiza la ficha de un empleado, por lo que no requiere de modificaciones externas iniciales.

#6 DEPARTMENTS: Se ha ampliado la lista de departamentos disponibles, duplicando algunos para asignarlos a cada región. Utiliza identificadores de localizaciones y empleados (el MANAGER_ID es un identificador de empleado) ya creados anteriormente. Cada departamento deberá tener un manager asociado.

En la ampliación de datos, es necesario conocer la estructura organizativa de partida, para poder insertar los nuevos puestos en la estructura jerárquica de forma coherente. A partir del organigrama original, se integran el resto de departamentos que

ya había creados pero no habían sido utilizados y los departamentos con los nuevos managers. Inicialmente se pueden crear departamentos sin asignar un manager.

#7 EMPLOYEES: es la última tabla a la que se le añaden datos porque es la que reúne más claves ajenas, por lo que necesita que todos los datos a los que hace referencia ya hayan sido creados. Esta tabla contendrá el grueso de los datos.

Primero se crean los managers y se asocian a los departamentos. También es necesario crearlos antes que al resto de empleados porque el campo MANAGER_ID de EMPLOYEES no puede hacer referencia a un empleado inexistente.

Finalmente, para dar coherencia interna a los datos, es necesario obtener qué puestos de trabajo están en qué departamento, así como separar los cargos de dirección de los que no lo son.

El esquema empleado, HR, está compuesto por siete tablas donde 'employees' es la central, como se puede observar en el siguiente esquema ER:

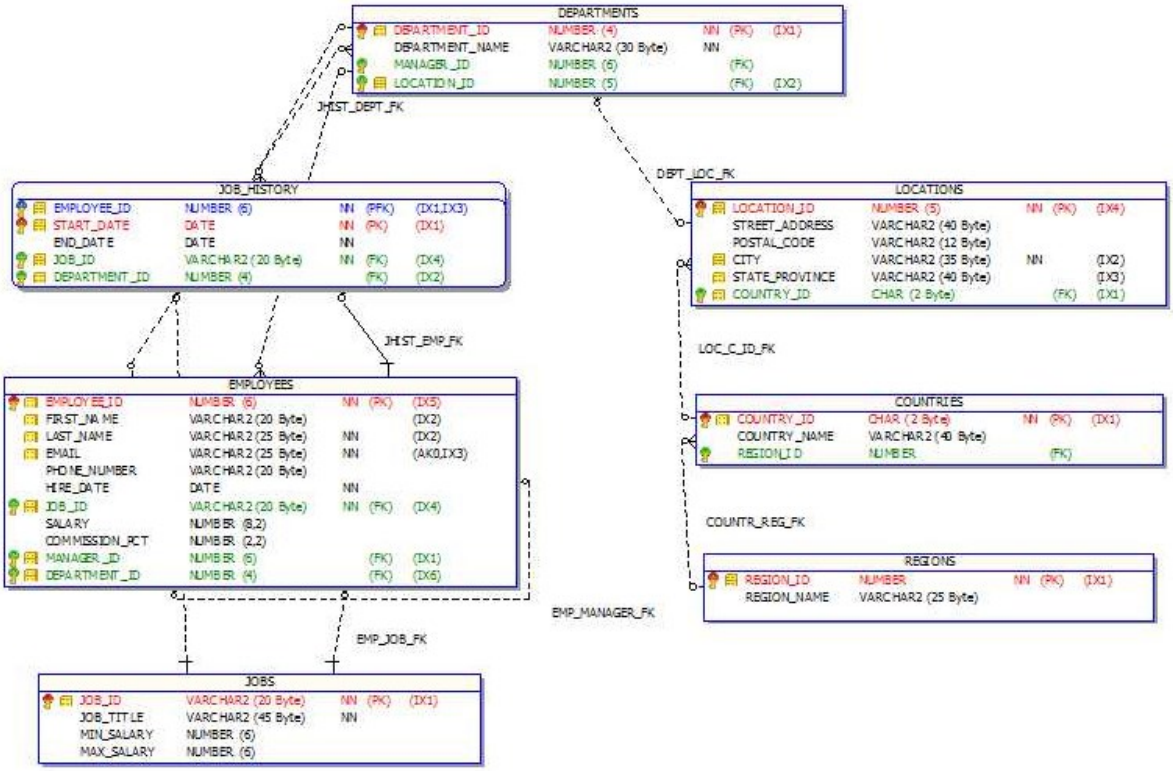


Ilustración 26: Diagrama ER del esquema HR. Fuente: TOAD

5.1.4 Diseño de consultas

Para analizar las herramientas SQL Developer y Toad se han elaborado tres consultas sobre el esquema HR que representen distintos niveles de complejidad para aportar diversidad al estudio. Partiendo de una misma consulta base, en las secciones dedicadas a SQL Developer [5.2 Optimización de consultas en SQL Developer] y TOAD [5.3 Optimización de consultas en TOAD] se describirán los resultados obtenidos a partir de la batería de pruebas diseñada [5.1 Diseño de los casos de uso].

La traducción de lenguaje natural a lenguaje SQL en las consultas se ha llevado a cabo mediante análisis manual de la consulta, dividiendo la consulta en subpartes e integrándolas a continuación entre sí.

El enunciado correspondiente a cada una de las consultas es el siguiente:

- Consulta 1: Dado un cargo de mando ('manager') similar, se requiere obtener el nombre de la región y los departamentos asignados a dichos jefes de departamento, así como el número de subordinados a su cargo que fueron contratados a partir de una determinada fecha. Todo ello ordenado ascendentemente en función del número de empleado de los jefes de departamento o '*managers*'.

- Consulta 2: Dado un puesto de trabajo similar entre regiones, mostrar por cada región: 1) Media del salario en esa región para ese puesto similar; 2) Salario máximo pagado en ese puesto actualmente (no el máximo permitido); 3) Salario mínimo pagado en ese puesto (no el mínimo permitido); 4) Número de empleados relativos al total de los que ocupan el mismo cargo que comparten puesto y región, y cuyo salario se encuentra por encima de la media; 5) Número de empleados relativos al total de los que ocupan el mismo cargo que comparten puesto y región, y cuyo salario se encuentra por debajo o igual a la media; 6) Para cada puesto concreto similar al indicado (utilizando una palabra clave de su nombre), mostrar el total de empleados que trabajan en él; 7) Ordenado ascendentemente por media salarial

- Consulta 3: Determinar en qué países, departamentos y puestos de trabajo se han llevado a cabo más contrataciones en los últimos X años, diciendo cuántas, y ordenando los resultados por orden descendente. Además, restringir a sólo los 10 puestos de trabajo con mayor número de contrataciones.

Consulta 1:

```
select q1.department_id, q1.department_name, q1.region_name, q3.manager_id, q3.subordinados

from

(select departments.department_id, departments.department_name, regions.region_name from departments join locations on departments.location_id=locations.location_id join countries on locations.country_id=countries.country_id join regions on countries.region_id=regions.region_id) q1,

(select distinct departments.department_id, departments.manager_id from departments join employees on departments.manager_id=employees.employee_id join jobs on employees.job_id=jobs.job_id where jobs.job_title like '%Finance%') q2,

(select employees.manager_id,count(*) as subordinados from departments join employees on departments.manager_id=employees.manager_id where employees.hire_date > '01/01/2000' group by employees.manager_id) q3

where q1.department_id=q2.department_id and q2.manager_id=q3.manager_id

order by q3.manager_id asc;
```

DEPARTMENT_ID	DEPARTMENT_NAME	REGION_NAME	MANAGER_ID	SUBORDINADOS
100	Finance_America	Americas	108	9
370	Finance_EMEA	Europe	233	15
540	Finance_Asia	Asia	250	20

Ilustración 27: Salida Consulta 1

Consulta 2:

```
select q1.region_name, q1.department_id, q2.job_id, q2.AVG_Sal_JobId,
q3.MAX_Sal_JobId,    q6.Mas_AVG,    q4.MIN_Sal_JobId,    q7.Menos_AVG,
q5.Total_JobId

from

      (select               distinct               regions.region_name,
departments.department_id,    jobs.job_id    from    jobs,    employees,
departments, locations, countries, regions where jobs.job_title like
'%Marketing%'        and        jobs.job_id=employees.job_id        and
employees.department_id=departments.department_id                   and
departments.location_id=locations.location_id                       and
locations.country_id=countries.country_id                           and
countries.region_id=regions.region_id) q1,

      (select distinct employees.job_id, avg(salary) as AVG_Sal_JobId
from  employees,  jobs  where  jobs.job_title  like  '%Marketing%'  and
jobs.job_id=employees.job_id group by employees.job_id) q2,

      (select distinct employees.job_id, max(salary) as MAX_Sal_JobId
from  employees,  jobs  where  jobs.job_title  like  '%Marketing%'  and
jobs.job_id=employees.job_id group by employees.job_id) q3,

      (select t1.job_id,count(*) as Mas_AVG

      from

            (select  job_id,  salary  from  employees  where  job_id  like
'%MK%') t1,

            (select distinct job_id, avg(salary) as AVG_Sal_JobId from
employees where job_id like '%MK%' group by job_id) t2

      where t1.job_id=t2.job_id and t1.salary > t2.AVG_Sal_JobId

      group by t1.job_id) q6,

      (select distinct employees.job_id, min(salary) as MIN_Sal_JobId
from  employees,  jobs  where  jobs.job_title  like  '%Marketing%'  and
jobs.job_id=employees.job_id group by employees.job_id) q4,

      (select r1.job_id, count(*) as Menos_AVG

      from

            (select  job_id,  salary  from  employees  where  job_id  like
'%MK%') r1,

            (select distinct job_id, avg(salary) as AVG_Sal_JobId from
employees where job_id like '%MK%' group by job_id) r2
```

```
      where r1.job_id=r2.job_id and (r1.salary < r2.AVG_Sal_JobId or
r1.salary = r2.AVG_Sal_JobId)

      group by r1.job_id) q7,

      (select distinct job_id, count(employee_id) as Total_JobId from
employees where job_id like '%MK%' group by job_id) q5

where     q1.job_id=q2.job_id     and     q1.job_id=q3.job_id     and
q1.job_id=q4.job_id  and  q1.job_id=q5.job_id  and  q1.job_id=q6.job_id
and q1.job_id=q7.job_id

order by q1.department_id, q2.job_id asc;
```

REGION_NAME	DEPARTMENT_ID	JOB_ID	AVG_SAL_JOBID	MAX_SAL_JOBID	MAS_AVG	MIN_SAL_JOBID	MENOS_AVG	TOTAL_JOBID
Americas	20	MK_MAN	11883,3333	13000	1	11000	2	3
Americas	20	MK_REP	5944,28571	8050	4	4060	3	7
Europe	290	MK_MAN_EMEA	13026,3636	16325	2	12435	9	11
Europe	290	MK_REP_EMEA	6570,46	8500	3	6400	47	50
Asia	460	MK_MAN_Asia	7246,6	10000	3	6915	22	25
Asia	460	MK_REP_Asia	5956,95833	6000	115	4500	5	120

6 rows selected.

Ilustración 28: Salida Consulta 2

Consulta 3:

```
select job_id, department_id, country_name, Empl_Contratados,
Total_JobId

from

      (select q1.job_id, q1.department_id, q1.country_name,
      q2.Empl_Contratados, q3.Total_JobId

      from

            (select distinct employees.job_id,
      employees.department_id, countries.country_name from employees,
      departments, locations, countries where
      employees.department_id=departments.department_id and
      departments.location_id = locations.location_id and
      locations.country_id = countries.country_id) q1,

            (select distinct job_id, count(employee_id) as
      Empl_Contratados from employees where substr(TO_CHAR(hire_date,
      'yyyy/mm/dd'),1,4) between (substr(TO_CHAR(SYSDATE,
      'yyyy/mm/dd'),1,4)-5) and (substr(TO_CHAR(SYSDATE,
      'yyyy/mm/dd'),1,4)) group by job_id) q2,

            (select distinct job_id, count(employee_id) as Total_JobId
      from employees group by job_id) q3

      where q1.job_id=q2.job_id and q1.job_id=q3.job_id

      order by q2.Empl_Contratados desc)

where rownum <= 10;
```

JOB_ID	DEPARTMENT_ID	COUNTRY_NAME	EMPL_CONTRATADOS	TOTAL_JOBID
MF_CLERK	170	United States of America	208	320
CONS_CLERK	180	United States of America	201	320
SA_REP_EMEA	350	United Kingdom	114	210
SA_REP_Asia	520	Australia	91	288
CT_CLERK_Asia	570	India	64	196
ST_CLERK_Asia	490	India	63	192
RET_CLERK	250	United States of America	63	96
GOV_CLERK	240	United States of America	62	100
IT_SUP_CLERK_EMEA	420	Ireland	62	100
IT_HELP_CLERK_EMEA	430	Ireland	59	100

10 rows selected.

Ilustración 29: Salida Consulta 3

5.1.5 Procedimientos

El procedimiento empleado para efectuar cada prueba varía según se trate de SQL Developer o TOAD:

- En SQL Developer:

Para cada consulta se aplica el mismo conjunto de pruebas, con el fin de asegurar la comparabilidad de resultados y conclusiones entre ellas. La batería de pruebas está compuesta de pruebas agrupadas según la influencia de qué parámetro se esté evaluando (sus significados serán explicados en [5.2 Optimización de consultas en SQL Developer]):

  - Las pruebas *cX_p0a* y *cX_p0b* están dedicadas a ejecutar la consulta original por duplicado, con dos fines 1) comprobar las mejoras en el rendimiento que tienen lugar cuando ya hay información en memoria (y ya no es necesario acceder al disco para obtenerla) y 2) establecer un punto de referencia, que es la segunda prueba, en la que ya se ha tenido en cuenta el efecto de los accesos a memoria antes que a disco.

  - La prueba *cX_p1* varía el nivel de las estadísticas recopiladas, subiendo el nivel de 'TYPICAL' a 'ALL'. No se utilizará en ningún caso 'BASIC' porque en este caso el optimizador no está capacitado para detectar cuándo una estadística está obsoleta, lo que supone un claro retroceso en la optimización de la consulta.

  - Las pruebas *cX_p1*, *cX_p2*, *cX_p3*, *cX_p4* y *cX_p5* se ejecutan combinando el nivel de estadísticas ('ALL' y 'TYPICAL') con los niveles del optimizador PL/SQL (controlado por el parámetro 'plsql_opt_level'), que varían entre 0 y 3, siendo 2 por defecto.

  - Las pruebas *cX_p6a* a *cX_p6*e se centran en los efectos que puedan tener los diferentes tamaños de muestreo, contenidos en el parámetro 'optimizer_dynamic_sampling'. Su valor máximo es 10, por lo que las pruebas están realizadas empleando los valores 2, 5, 8 y 10. Por defecto es 2.

  - Las pruebas *cX_p7a* y *cX_p7b* exploran los efectos del optimizador en modo 'CHOOSE', mientras que *cX_p11a* y *cX_p11b* utilizan el modo 'FIRST_ROWS'. El modo por defecto empleado es 'ALL_ROWS'.

  - Las pruebas *cX_p8a* a *cX_p8f* combinan los efectos de los distintos tamaños fetch (50 - por defecto, 100, 150, 200) con los tamaños de muestreo 2 y 8.

  - El grupo de pruebas *cX_p9* está dedicado a las ejecuciones que involucran variables bind o de enlace, si pudieran usarse en la consulta.

  - El grupo de pruebas *cX_p10* utilizan la vista 'emp_details_view' y sus variantes incluyendo variables bind.

- Existen otros grupos específicos de cada consulta debido a la variedad buscada en los casos de uso, que permitieran evaluar distintos aspectos y complejidades.

Por cada ejecución de las pruebas expuestas se repite el siguiente procedimiento para extraer los datos de las fuentes empleadas:

1- Volver al estado de referencia inicial, para asegurar que los resultados no son fruto de una información precedente sino que son por una mejora real en la consulta. Para ello existen dos posibilidades: 1) Cerrar la sesión actual y abrir una nueva, o 2) Modificar cada uno de los parámetros. En este caso, se ha optado por la primera opción combinado con la ejecución de los siguientes comandos:

En las preferencias de SQL Developer, se mantiene desactivada la opción "Guardar variables de enlace en disco al salir" ("*Save bind variables to disk on exit*"). Si se activara esta opción, las ejecuciones de las mejoras perderían independencia porque ya tendrían información relacionada previa.

2- Cambiar para cada ejecución (cada adaptación de la consulta) el nombre del trace file, siguiendo la siguiente nomenclatura: cX_pY; siendo X el número de la consulta (1, 2 o 3) e Y el número de prueba. Y será 0 para la ejecución de partida, que servirá de referente para el resto de pruebas (de 1 en adelante).

3- Extraer los datos de las fuentes [Tabla 12: Fuentes de extracción de datos SQL Developer]

- En TOAD:

El procedimiento está basado en nueve tipos distintos de pruebas, que parten de una situación inicial en la que se ejecuta dos veces la consulta original (*cX_p0a_toad* y *cX_p0b_toad*) por dos motivos: 1) averiguar cuál es el rendimiento original que se desea mejorar, y 2) comprobar si existen mejoras en el rendimiento en la segunda ejecución debido al efecto de la memoria sobre el disco (es de esperar que en la primera existan lecturas de disco, y en cambio en la segunda toda la información haya sido ya insertada en memoria).

El siguiente subgrupo de pruebas, que contiene las pruebas *cX_p3_toad*, *cX_p5_toad*, *cX_p7_toad* y *cX_p8_toad*, buscará alternativas en base a una combinación específica de los niveles de Optimizer Intelligence e Index Generator Intelligence [4.2 Herramientas] de la función de optimización Advance SQL Optimizer (un ejemplo de uso de esta función ha sido incluido en la sección 9.4 Propuesta de guía de trabajo en TOAD):

Prueba	Optimizer Intelligence Level (1 - 5)	Nivel Index Generation Intelligence Level (1 - 5)
c1_p3_toad	2	2
c1_p5_toad	5	2
c1_p7_toad	2	5
c1_p8_toad	5	5

Tabla 11: TOAD - Session Settings: OptimizerIL e IndexIL

Las pruebas *cX_p4_toad* emplearán la vía de optimización basada en Oracle: OEM.

El tercer método de optimización posible, Auto Optimize SQL, será empleado en las pruebas *cX_p6_toad*, buscando como en las anteriores dos alternativas según los criterios de mejor tiempo y menor coste.

Las mejoras encontradas en cada una de las pruebas y según cada criterio correspondiente han sido incluidas en los anexos [9.3 Alternativas de las consultas TOAD] con el fin de ofrecer visión de las mejoras completa, y no basarla únicamente en la exposición y descripción de los resultados.

5.2 Optimización de consultas en SQL Developer

Las mejoras aplicadas sobre las consultas están esencialmente relacionadas con el uso de vistas, atajos o preferencias (*hints*) y reestructuración del código. Sin embargo, conviene tener en cuenta que estas mejoras están aplicadas dentro de un contexto concreto, de forma que pueden no suponer una mejora válida en otras consultas.

La mayor fuente de información de información sobre la ejecución de las sentencias SQL son el Explain Plan, el plan de ejecución y las estadísticas de ejecución [4.1 Mecanismos]. La recolección de la información necesaria para evaluar cada mejora se ha basado esencialmente en la modificación de los siguientes parámetros de la base de datos:

`sql > set timing on; //` Para mostrar el tiempo que ha pasado ejecutando una orden

`sql > set wrap off; y set linesize xxxx; //` Para que los resultados no se trunquen y cada fila lógica se corresponda con una fila en la pantalla (en los casos en los que corresponda hacer esta modificación).

`sql > set autotrace on;` // Para activar la función de trazabilidad en SQL*Plus, y que por cada operación ejecutada se muestre el plan de ejecución y sus correspondientes estadísticas. Después de activarlo, el parámetro "sql_trace" debe figurar con valor "true". Inicialmente fue empleado como primera aproximación, pero fue sustituido por la función autotrace de SQL Developer, que garantizaba resultados reales y no estimados.

`sql > set user_dump_dest …` // Para definir un nuevo trace file por cada prueba ejecutada. Esto permite aislar cada prueba del conjunto.

`sql > set statistics_level [TYPICAL || ALL] //` Define el nivel de detalle de las estadísticas que se recopilen. Se desaconseja establecer el nivel 'ALL' para toda la base de datos porque conllevaría un gran consumo de recursos.

`sql > set timed_statistics TRUE;` // Para activar la recopilación automática de estadísticas.

```
sql > optimizer_use_sql_plan_baselines TRUE;
```

```
sql > optimizer_capture_sql_plan_baselines FALSE;
```

`sql > optimizer_dynamic_sampling 2;` // Es el valor por defecto del parámetro.

```
sql > optimizer_features_enable 11.2.0.2;
```

`sql > optimizer_mode ALL_ROWS;` // Tiene otros modos: FIRST_ROWS Y CHOOSE.

Teniendo en cuenta esta información sobre SQL Developer, se presentan las mejoras respectivas de cada consulta primero enumerando qué acciones se han tomado para mejorarla y a continuación mostrando en una tabla el valor de los indicadores seleccionados para comparar mejoras.

Estos indicadores son el coste, tiempo real y tiempo pasado, número de lecturas lógicas y físicas, número estimado de líneas y número real de líneas extraídas (no se consideran operaciones de escritura ya que no se han empleado sentencias 'insert' o 'update', es decir, no hay modificación de datos, sólo lectuas). Aunque existen otros parámetros disponibles para valorar cada versión de la consulta, se han escogido aquellos más representativos.

En la siguiente tabla se describen cada uno de los parámetros considerados en el análisis de las consultas así como las fuentes de las cuales se extraen sus valores:

Parámetro	*Fuente(s)*	*Descripción*
Coste (%CPU)	Autotrace SQL Developer (‘Cost’) y XplanSQLDev (‘Cost (%CPU)’)	Coste de la operación para la CPU
E-Rows	Autotrace SQL Developer (‘Cardinality’) y XplanSQLDev (‘E-Rows’)	Líneas estimadas que se extraerían.
A-Rows	Autotrace SQL Developer (‘Last_Output_Rows’), XplanSQLDev (‘A-Rows’) y Tracefile + TKPROF (‘rows’)	Número real de líneas extraídas.
E-Time	XplanSQLDev (‘E-Time’)	Tiempo estimado de ejecución
A-Time	XplanSQLDev (‘A-Time’)	Tiempo real de ejecución
LRs	Autotrace SQL Developer (‘Last_CR_Buffer_Gets’), XplanSQLDev (‘Buffers’) y Tracefile + TKPROF (‘query’)	Número de lecturas lógicas
PRs	Autotrace SQL Developer (‘Last_Disk_Gets’), XplanSQLDev (‘Reads’) y Tracefile + TKPROF (‘disk’)	Número de lecturas físicas
Optimizador	Autotrace SQL Developer (‘Optimizer’) y Tracefile + TKPROF (‘Optimizer_mode’)	Modo en que se ha trabajado el optimizador.

Tabla 12: Fuentes de extracción de datos SQL Developer

Se ha incluido en los anexos [9.2 Comandos de ejecución útiles], un ejemplo del proceso seguido para extraer los datos indicados de las tres fuentes para una misma consulta.

Como se ha visto en [5.1 Diseño de los casos de uso], subapartado ‘Procedimientos’, las pruebas en SQL Developer giran en torno a la combinación de los siguientes parámetros y elementos:

- Nivel de recopilación de estadísticas: TYPICAL (por defecto) y ALL.
- Tamaño de muestreo dinámico (dynamic sampling): 2 (por defecto), 5, 8 o 10. En combinaciones avanzadas se reducen las pruebas a 2 y 8. Este parámetro es modificable tanto a nivel de sesión como de sistema, y su valor (entre 0 y 10) establece cuándo recoger estadísticas dinámicas y el tamaño de la muestra que el optimizador utiliza para ejecutar dichas estadísticas. Su valor por defecto depende de la versión Oracle (‘optimizer_features_enable’) que se esté empleando, siendo 2 si es superior a la 10, 1 si es la 9.2.0 o 0 si es inferior a la 9.2.0. El muestreo en la recopilación de estadísticas es importante porque llevar a cabo este proceso sin muestra conllevaría full scans y sorts (ordenaciones) de las tablas completas, incluso en las que no sería lo mejor. El muestreo permite minimizar el uso de recursos al recolectar estadísticas. Se probará en intervalos de 3: 2, 5, 8 y 10. Por otro lado, existe la posibilidad de permitir al optimizador

escoger el tamaño adecuado de la muestra ejecutando "`execute dbms_stats.gather_schema_stats ('HR',dbms_stats.auto_sample_size);`".

- Nivel del optimizador PL/SQL (por GUI: Preferencias > Base de datos): 0, 1, 2 (por defecto) o 3.

- Utilización de la vista 'emp_details_view', que concentra la información más relevante de la base de datos en una sola vista y por ello su tamaño es sensiblemente superior al de las tablas de la base de datos:

```
SQL> describe emp_details_view;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 EMPLOYEE_ID                               NOT NULL NUMBER(6)
 JOB_ID                                    NOT NULL VARCHAR2(20)
 MANAGER_ID                                         NUMBER(6)
 DEPARTMENT_ID                                      NUMBER(4)
 LOCATION_ID                                        NUMBER(5)
 COUNTRY_ID                                         CHAR(2)
 FIRST_NAME                                         VARCHAR2(20)
 LAST_NAME                                 NOT NULL VARCHAR2(25)
 SALARY                                             NUMBER(8,2)
 COMMISSION_PCT                                     NUMBER(2,2)
 DEPARTMENT_NAME                           NOT NULL VARCHAR2(30)
 JOB_TITLE                                 NOT NULL VARCHAR2(45)
 CITY                                      NOT NULL VARCHAR2(35)
 STATE_PROVINCE                                     VARCHAR2(40)
 COUNTRY_NAME                                       VARCHAR2(40)
 REGION_NAME                                        VARCHAR2(25)
```

Ilustración 30: Descripción de la vista 'emp_details_view'

- Bind variables o variables de enlace. Se aplican para sustituir aquellas partes de la consulta que pudieran tomar otros valores. Por ejemplo, el nombre del puesto (job_title) y la fecha de contratación (hire_date).

- Fetch rows o tamaño de recuperación de datos: está comprendido entre 50 y 200, y establece la cantidad de información que puede extraerse en una sola transacción. Un número muy bajo conllevará un elevado número de transacciones, mientras que uno muy alto agotará antes el espacio disponible en la caché y a la larga requerirá más transacciones y más tiempo en cada transacción (extraer más información conlleva más tiempo). En este caso se prueba con intervalos de 50, es decir: 50 (por defecto), 100, 150 y 200.

Consulta 1:

La primera ejecución (*c1_p0a*) mantiene los parámetros en su valor por defecto, lo que significa que el nivel de las estadísticas es 'typical', el nivel del optimizador PL/SQL es 2, el modo es 'ALL_ROWS', el tamaño de muestreo es 2, el nivel de fetch rows es 50, y no se utilizan variables bind ni hints (el hint 'result_cache' no está disponible para a versión exprés. Como se esperaba, esta primera ejecución sí ha necesitado realizar accesos a disco para leer los datos, lo que se ha traducido en 85 lecturas físicas.

Las subsecuentes ejecuciones ya no precisan del disco para acceder a los datos, porque estos ya han sido llevados a memoria. Esto tiene dos efectos: 1) no habrá más lecturas físicas (PRs), sólo lógicas (LRs), y 2) el tiempo de ejecución se reduce considerablemente.

Como se puede observar en [**¡Error! No se encuentra el origen de la referencia.**], los efectos de los cambios en el nivel de las estadísticas no son apreciables (prueba *c1_p1*), por lo que se decide mantener el nivel típico con el fin de no sobrecargar la base de datos. Tampoco se aprecian cambios debidos a la variación del nivel del optimizador PL/SQL (pruebas *c1_p2* a *c1_p5*), por lo que se decide fijar su valor a 2 (el valor por defecto) para las restantes pruebas.

Por el contrario, sí se aprecian mejoras en cuanto al coste ante aumentos en el parámetro del tamaño de muestreo (grupo *c1_p6* de pruebas). Un mayor tamaño de muestreo disminuye el coste de ejecución, que pasa de 49 a 39 cuando este tamaño es 5, 8 o 10. Dentro de este mismo grupo, las pruebas *c1_p6d* y *c1_p6e* hacen uso de la función 'auto_sample_size' del paquete dbms_stats, según la cual el optimizador interno de Oracle debería determinar internamente cuál es el tamaño de muestreo adecuado para este caso concreto. Volviendo a establecer el valor por defecto del tamaño de muestreo (2) y ejecutando esta función, el coste de ejecución vuelve a niveles anteriores, lo que supone un empeoramiento. Sin embargo, si se aumenta el tamaño de muestreo a 8 y se ejecuta la función de dbms_stats, el coste vuelve a descender. Esto parece indicar que para la versión exprés el dimensionamiento automático del muestreo no está activado, aunque al ejecutar no muestre signos de error.

Este mismo comportamiento se repite en las pruebas *c1_p7a*, *c1_p7b*, *c1_p11a* y *c1_p11b*, en las que se utilizan los modos 'choose' y 'first_rows' del optimizador (el modo 'auto' no está permitido, sí lo está 'rule', pero es altamente desaconsejable utilizarlo, por lo que se omite). Si se restaura el valor por defecto del 'dynamic_sampling' a 2, la situación vuelve a ser la inicial, con coste 49. En cambio, cuando este parámetro toma valor 8, el coste disminuye a 39. El grupo de pruebas *c1_p8* sufre el mismo efecto, independientemente de los cambios en el parámetro fetch rows, y en *c1_p9*, con las variables bind.

En este último grupo, *c1_p9*, no es posible efectuar las pruebas en las cuales se trata de sustituir la variable 'hire_date' por una bind, ya que las variables DATE no están soportadas por las bind.

Hay un cambio significativo en el grupo de pruebas *c1_p10*, debido a que la consulta ha sido reescrita para utilizar la vista 'emp_details_view'. Las dos primeras, *c1_p10a* y *c1_p10b*, sólo evalúan el rendimiento de la consulta al utilizar esta vista y combinarla con los valores de muestreo 2 y 8. El mismo comportamiento se repite: cuando el valor del muestreo es 8, el coste disminuye, en caso contrario aumenta. Las dos restates pruebas, *c1_p10c* y *c1_p10d*, combinan la vista 'emp_details_view' con el uso de variables bind para el campo 'job_title' y los valores de muestreo 2 y 8. Los resultados son similares a las dos primeras, por lo que 1) el uso de variables bind no ha sido exitoso en este caso, y 2) sólo los cambios en el tamaño de muestreo parecen influir en el rendimiento de la consulta.

Con vistas se aumenta el número de operaciones (de 30 a 33), y además se cambia el plan de ejecución cuando el muestreo es 2 u 8. Según las comparaciones de v$mystat, en *c1_p10d* no es necesario realizar un index fast full scan, mientras que en *c1_p10c* se aplican tres. En *c1_p10d* hay además más hits en la caché del cursor (4) que en el *c1_p10c* (2). En cambio en *c1_p10c* hay menos sorts, tanto en memoria (3 frente a 6 del *c1_p10d*) como en cuanto al número de filas (1227 frente a 1490), ninguno requiere sorts en disco. El *c1_p 10d* ha tenido menos operaciones en las que haya recuperado un gran número de filas (>3), mientras que en el *c1_p 10c* aproximadamente el doble de operaciones lo han hecho. Esto queda reflejado en un sustancial cambio del coste (de 64 en el *c1_p 10c* a 41 en el *c1_p 10d*). La versión de la consulta 1 que utiliza la vista 'emp_details_view' quedaría tal que:

```
select q1.department_id, q1.department_name, q1.region_name, q3.manager_id,
q3.subordinados

from

      (select distinct department_id, department_name, region_name from
emp_details_view) q1,

      (select distinct departments.department_id, departments.manager_id
from departments join employees on departments.manager_id =
employees.employee_id join jobs on employees.job_id = jobs.job_id where
jobs.job_title like '%Finance%') q2,

      (select employees.manager_id,count(*) as subordinados from departments
join employees on departments.manager_id = employees.manager_id where
employees.hire_date > '01/01/2000' group by employees.manager_id) q3

where q1.department_id=q2.department_id and q2.manager_id=q3.manager_id

order by q3.manager_id asc;
```

A continuación se presentan en sendas tablas la descripción de las pruebas que se han elaborado para la primera consulta en la que se pueden observar los cambios en los parámetros y otra tabla que presenta los resultados de cada ejecución:

Consulta	ID_mejora	Archivo	Descripción	Statistics_level	plsql_optimize_level (SQL Dev GUI)	Optimizer_mode (cmd / SQL dev)	Opt_dynamic_sampling (v$parameters)	Bind variable	Fetch rows (SQL Dev GUI)
1	0a	c1_p0a	Ejecución inicial	TYPICAL	2	ALL_ROWS	2	X	50
1	0b	c1_p0b	Reejecución (se espera mejor resultado)	TYPICAL	2	ALL_ROWS	2	x	50
1	1	c1_p1	alter session set statistics_level='ALL'	ALL	2	ALL_ROWS	2	x	50
1	2	c1_p2	plsql_opt_level = 0 (GUI)	ALL	0	ALL_ROWS	2	x	50
1	3	c1_p3	plsql_opt_level = 1 (GUI)	ALL	1	ALL_ROWS	2	x	50
1	4	c1_p4	plsql_opt_level = 3 (GUI)	ALL	3	ALL_ROWS	2	x	50
1	5	c1_p5	0b + 2	TYPICAL	0	ALL_ROWS	2	x	50
1	6a	c1_p6a	Sampling = 5	TYPICAL	2	ALL_ROWS	5	x	50
1	6b	c1_p6b	Sampling = 8	TYPICAL	2	ALL_ROWS	8	x	50
1	6c	c1_p6c	Sampling = 10	TYPICAL	2	ALL_ROWS	10	x	50
1	6d	c1_p6d	execute dbms_stats.gather_schema_stats ('HR', dbms_stats.auto_sample_size);	TYPICAL	2	ALL_ROWS	2 (default), debería usar el auto	x	50
1	6e	c1_p6e	execute dbms_stats.gather_schema_stats ('HR', dbms_stats.auto_sample_size);	TYPICAL	2	ALL_ROWS	8, debería usar el auto	x	50
1	7a	c1_p7a	opt_mode = choose	TYPICAL	2	CHOOSE	2	x	50
1	7b	c1_p7b	opt_mode = choose	TYPICAL	2	CHOOSE	8	x	50
1	8a	c1_p8a	fetch = 100	TYPICAL	2	ALL_ROWS	2	x	100
1	8b	c1_p8b	fetch = 100	TYPICAL	2	ALL_ROWS	8	x	100
1	8c	c1_p8c	fetch = 150	TYPICAL	2	ALL_ROWS	2	x	150
1	8d	c1_p8d	fetch = 150	TYPICAL	2	ALL_ROWS	8	x	150
1	8e	c1_p8e	fetch = 200	TYPICAL	2	ALL_ROWS	2	x	200
1	8f	c1_p8f	fetch = 200	TYPICAL	2	ALL_ROWS	8	x	200
1	9a	c1_p9a	Bind: job_title (1 valor)	TYPICAL	2	ALL_ROWS	2	Job_title --> trabajo	50
1	9b	c1_p9b	Bind: job_title (1 valor)	TYPICAL	2	ALL_ROWS	8	Job_title --> trabajo	50
1	9	c1_p9	Bind: job_title (varios valores)	TYPICAL	2	ALL_ROWS	2	Job_title --> trabajo	50
1	9c	c1_p9c	Bind: hire_date	TYPICAL	2	ALL_ROWS	2	hire_date --> contratacion	50
1	9d	c1_p9d	Bind: hire_date	TYPICAL	2	ALL_ROWS	8	hire_date --> contratacion	50
1	10a	c1_p10a	Vista emp_details_view	TYPICAL	2	ALL_ROWS	2	X	50
1	10b	c1_p10b	Vista emp_details_view	TYPICAL	2	ALL_ROWS	8	X	50
1	10c	c1_p10c	Vista emp_details_view + bind job_title	TYPICAL	2	ALL_ROWS	2	Job_title --> trabajo	50
1	10d	c1_p10d	Vista emp_details_view + bind job_title	TYPICAL	2	ALL_ROWS	8	Job_title --> trabajo	50
1	11a	c1_p11a	opt_mode = first_rows	TYPICAL	2	FIRST_ROWS	2	x	50
1	11b	c1_p11b	opt_mode = first_rows	TYPICAL	2	FIRST_ROWS	8	x	50
1	11c	c1_p11c	opt_mode = auto	TYPICAL	2	AUTO	2	x	50
1	11d	c1_p11d	opt_mode = auto	TYPICAL	2	AUTO	8	x	50

Tabla 13: SQL Developer - Descripción pruebas Consulta 1

Consulta	ID_mejora	Archivo	Coste (%CPU)	E-Rows	A-Rows	E-Time	A-Time	LRs	PRs	Optimizador
1	0a	c1_p0a	49 (100%)	1	3	0:00:01	00:00:00.08	123	85	ALL_ROWS
1	0b	c1_p0b	49 (100%)	1	3	0:00:01	00:00:00.01	123	0	ALL_ROWS
1	1	c1_p1	49 (100%)	1	3	0:00:01	00:00:00.01	123	0	ALL_ROWS
1	2	c1_p2	49 (100%)	1	3	0:00:01	00:00:00.01	123	0	ALL_ROWS
1	3	c1_p3	49 (100%)	1	3	0:00:01	00:00:00.01	123	0	ALL_ROWS
1	4	c1_p4	49 (100%)	1	3	0:00:01	00:00:00.01	123	0	ALL_ROWS
1	5	c1_p5	49 (100%)	1	3	0:00:01	00:00:00.01	123	0	ALL_ROWS
1	6a	c1_p6a	39 (100%)	1	3	0:00:01	00:00:00.01	123	0	ALL_ROWS
1	6b	c1_p6b	39 (100%)	1	3	0:00:01	00:00:00.01	123	0	ALL_ROWS
1	6c	c1_p6c	39 (100%)	1	3	0:00:01	00:00:00.01	123	0	ALL_ROWS
1	6d	c1_p6d	49 (100%)	1	3	0:00:01	00:00:00.01	123	0	ALL_ROWS
1	6e	c1_p6e	39 (100%)	1	3	0:00:01	00:00:00.01	123	0	ALL_ROWS
1	7a	c1_p7a	49 (100%)	1	3	0:00:01	00:00:00.01	123	0	CHOOSE
1	7b	c1_p7b	39 (100%)	1	3	0:00:01	00:00:00.01	123	0	CHOOSE
1	8a	c1_p8a	49 (100%)	1	3	0:00:01	00:00:00.01	123	0	ALL_ROWS
1	8b	c1_p8b	39 (100%)	1	3	0:00:01	00:00:00.01	123	0	ALL_ROWS
1	8c	c1_p8c	49 (100%)	1	3	0:00:01	00:00:00.01	123	0	ALL_ROWS
1	8d	c1_p8d	39 (100%)	1	3	0:00:01	00:00:00.01	123	0	ALL_ROWS
1	8e	c1_p8e	49 (100%)	1	3	0:00:01	00:00:00.01	123	0	ALL_ROWS
1	8f	c1_p8f	39 (100%)	1	3	0:00:01	00:00:00.01	123	0	ALL_ROWS
1	9a	c1_p9a	49 (100%)	1	3	0:00:01	00:00:00.01	123	0	ALL_ROWS
1	9b	c1_p9b	39 (100%)	1	3	0:00:01	00:00:00.01	123	0	ALL_ROWS
1	9c	c1_p9c								
1	9d	c1_p9d								
1	10a	c1_p10a	64 (100%)	51	3	0:00:01	00:00:00.02	152	0	ALL_ROWS
1	10b	c1_p10b	41 (100%)	26	3	0:00:01	00:00:00.01	129	0	ALL_ROWS
1	10c	c1_p10c	64 (100%)	51	3	0:00:01	00:00:00.03	152	0	ALL_ROWS
1	10d	c1_p10d	41 (100%)	26	3	0:00:01	00:00:00.01	129	0	ALL_ROWS
1	11a	c1_p11a	49 (100%)	1	3	0:00:01	00:00:00.01	123	0	FIRST_ROWS
1	11b	c1_p11b	39 (100%)	1	3	0:00:01	00:00:00.01	123	0	FIRST_ROWS
1	11c	c1_p11c								AUTO
1	11d	c1_p11d								AUTO

Tabla 14: SQL Developer - Resultados Consulta 1

Consulta 2:

A raíz de lo experimentado en la consulta 1, se han eliminado de la batería de pruebas aquellas que involucran variables bind sobre variables de tipo 'date' (fecha) o métodos del optimizador distintos de 'all_rows', 'first_rows' y 'choose'.

Como ya se observó en la consulta 1, la segunda ejecución, *c2_p0b*, elimina cualquier lectura física que tuviera lugar en *c2_p0a* al contar con la información en memoria y no ser necesario acudir al disco. Si en la primera se daban 121 PRs, en la segunda y las siguientes ejecuciones, sólo hay LRs.

La prueba *c2_p1*, en la que se utiliza el nivel de estadísticas 'all', no afecta a la ejecución de esta consulta. Sin embargo, es sabido por la documentación de Oracle que este nivel, al ser el más detallado en la recopilación de estadísticas, puede afectar negativamente al rendimiento si se aplica a toda la base de datos (no sólo a una sesión concreta) o sobre una base de datos más pesada, por lo que se adopta 'typical' como el nivel por defecto de las restantes pruebas.

En el grupo de pruebas dedicaso a evaluar el impacto de los niveles del optimizador PL/SQL (*c2_p2* a *c2_p5*), aunque no hay diferencias significativas, se aprecian mejoras leves cuando plsql_optimize_level = 1 en vez de 0 en las estadísticas

comparadas de v$mystat. Por ejemplo, los parámetros ‘CPU used when called started’ = 31 vs 29 y ‘Parse time elapsed’ = 23 vs 22 demuestran cierta mejoría entre los niveles.

Dentro del grupo *c2_p6*, centrado en evaluar la influencia de los tamaños de muestreo, se han dado los siguientes hechos:

- En c2_p6a, donde se aumenta el tamaño de muestreo a 5, el número de 'session logical reads' (lecturas lógicas) es muy superior respecto a 00b: 642 en 00b vs 2779 en 6a. A cambio, el aumento del tamaño muestral ha consumido más cpu y ha conllevado más operaciones de tipo hard parse.

- Entre 6a y 6b, hay en 6b un uso levemente superior de los recursos, pero no remarcable, debido al aumento del tamaño de muestreo.

- Se repite la tendencia: cuanto mayor es el tamaño de muestreo más CPU se consume y más tiempo requieren las operaciones parse, pero esto no se traduce en una mejor estimación, sino que se mantiene.

- En 6d, respecto a 00b se produce una mejora debida a la recopilación de estadísticas antes de la ejecución, dado que el resto de parámetros se ha mantenido sin cambios. Aunque la mejora es leve, en entornos más exigentes podría ser más útil. Tanto la cantidad de CPU usada (29 vs 25) como el tiempo de parse (22 vs 19) en v$mystat así lo demuestran.

- En 6e, respecto al 6b (con quien comparte la misma configuración de parámetros, pero sin estadísticas forzadas previas), no hay la suficiente diferencia como para considerarla mejor o peor opción. Respecto a 6d sí que se ha producido un empeoramiento, los tiempos de parse y consumo de CPU han aumentado.

Para las pruebas *c2_p7* se percibe una leve desmejoría cuando el modo del optimizador usado es ‘choose’:

- En *c2_p7a* se aprecia un leve sobrecoste respecto a 00b. Por ejemplo, se produce una ordenación en memoria más (según v$mystat) así como más consumo de CPU (aunque no significativo). En cambio, respecto a 6d (en la que se recopilaban estadísticas y se utilizaba un tamaño de muestreo = 2) sí que se observa un peor rendimiento según v$mystat: el tiempo consumido por la CPU es mayor (25 vs 31) y también el tiempo dedicado a parse (19 vs 24). Esta diferencia está causada especialmente por la recopilación previa de estadísticas actualizadas, que permiten diseñar un plan de ejecución mejor adaptado.

- En *c2_p7b*, respecto a 6b se aprecia una leve desmejora, visible en el aumento de CPU empleada y una ordenación (sort) en memoria más, pero no son datos destacables. Si se compara con 6e, la conclusión es similar.

Las pruebas *c2_p8* sobre el tamaño de fetch rows han mostrado los siguientes hechos:

- En *c2_p8a*, en comparación a la ejecución 00b, no hay mejoras significativas en cuanto a rendimiento que sitúen a una de las dos opciones claramente mejor que la otra. Sin embargo, sí se ha observado una gran mejora en cuanto al número de llamadas hechas por el usuario ('user calls' en v$mystat), que ha pasado de 16 a 10, ya que en este caso se recupera más información unitariamente.

- En c2_p8b y respecto a 00b, el aumento del consumo de CPU es apreciable (29 vs 47) así como el tiempo dedicado en tareas de parse (21 vs 39). En cambio el número de llamadas de usuario vuelve a decrecer (16 vs 10).

- Al igual que en 8b, las variaciones en cuanto a rendimiento no son determinantes (son muy leves) pero sí hay una disminución en el parámetro 'user calls' de v$mystat entre c2_p00b y c2_p8c, que pasa de 16 a 8.

- Para la prueba c2_p8d, en contraste con la ejecución 00b se produce un incremento en el uso de la CPU, de 29 a 47, y el tiempo dedicado al parse, de 21 a 41; pero menos llamadas de usuario (16 a 8). En 8e y 8f la relación de parámetros respecto a 00b es similar.

Las pruebas *c2_p9*, que incluyen el uso de variables bind en la mejora de la consulta, muestran que:

- En base a los datos de las vistas v$mystats de las ejecuciones 00b y 9a, el rendimiento sería similar en términos de uso de CPU. En la 9a sin embargo, a pesar de haber tareas de parse, no figura tiempo alguno. Esto es debido a que una operación parse involucra crear un explain plan, lo que es incompatible con las bind, por lo tanto lo ignora.

- Siguiendo los valores de v$mystat, el consumo de CPU es menor en 9b que en 00b (29 vs 10), y necesita menos tiempo, ya que de nuevo los tiempos asociados al parse son 0.

Para la consulta 2, el empleo de la vista ‘emp_details_view’ es claramente una mala elección, supone mayor consumo de CPU respecto a la ejecución de partida 00b, pasando de 29 a 299. Este comportamiento se repite en cuanto a tiempos de ejecución y de parse. Sin embargo el número de llamadas es el mismo, gracias a que el tamaño de la captura es superior. Este bajo rendimiento se produce también en 10b, por lo que se descarta realizar pruebas que combinen la vista y variables bind.

Al cambiar el modo del optimizador a ‘first_rows’, la diferencia observada responde a cambios en el tamaño de muestreo y no a cambios en el optimizador. Por un lado, si se emplea un muestreo = 2, respecto a 00b en 11a hay un uso levemente superior de la CPU, de 29 a 34, y también de los tiempos de parse, pero en general no hay una diferencia remarcable que indique una opción claramente superior. Por otro

lado, si se utiliza un tamaño de muestreo = 8 la diferencia de uso de CPU entre 00b y 11b es más acusada, pasa de 29 a 45, y el tiempo de parse también se ve afectado: 22 vs 37.

Finalmente, en esta consulta se ha podido llevar a cabo una mejora adicional (pruebas *c2p12*), que consiste en reestructurar la sentencia SQL para unificar secciones que compartan los mismos accesos con el fin de minimizar el número de operaciones y transacciones.

- Utilizando un muestreo = 2 (*c2_p12a*), los tiempos son más reducidos en comparación con 00b, y el execution plan contiene menos operaciones; en el mismo número de llamadas de usuario se extraen más datos. En conjunto, es más recomendable.

- Si se aumenta el muestreo a 8 (*c2_p12b*), se emplea más CPU que en 00b, 29 vs 50; y también se dedica más tiempo a las tareas de parse (21 vs 44). Atendiendo al valor extraído de los informes, seguiría siendo más recomendable que 00b. En comparación con 12a, utilizando las estadísticas de v$mystat, sería más óptima la versión de la ejecución 12a.

A continuación se muestran las tablas homólogas a las presentadas para la consulta 1, primero una tabla con la descripión de la configuración de cada prueba y a continuación la tabla contenedora de los resultados.

Tabla 15: SQL Developer - Descripción pruebas Consulta 2

Consulta	ID_mejora	Archivo	Descripción	Statistics_level	plsql_optimize_level (SQL Dev GUI)	Optimizer_mode (cmd / SQL dev)	Opt_dynamic_sampling (v$parameters)	Bind variable	Fetch rows (SQL Dev GUI)	Result_cache_mode
2	0a	c2_p0a; c2_p00	Ejecución inicial	TYPICAL	2	ALL_ROWS	2	X	50	x
2	0b	c2_p0b; c2_p00b	Reejecución	TYPICAL	2	ALL_ROWS	2	x	50	x
2	1	c2_p1	alter session set statistics_level='ALL'	ALL	2	ALL_ROWS	2	x	50	x
2	1	c2_p1b	alter session set statistics_level='ALL'	ALL	2; ALL = ENABLE (GUI)	ALL_ROWS	2	x	50	x
2	2	c2_p2	plsql_opt_level = 0 (GUI)	ALL	0	ALL_ROWS	2	x	50	x
2	3	c2_p3	plsql_opt_level = 1 (GUI)	ALL	1	ALL_ROWS	2	x	50	x
2	4	c2_p4	plsql_opt_level = 3 (GUI)	ALL	3	ALL_ROWS	2	x	50	x
2	5	c2_p5	0b + 2	TYPICAL	0	ALL_ROWS	2	x	50	x
2	6a	c2_p6a	Sampling = 5	TYPICAL	2	ALL_ROWS	5	x	50	x
2	6b	c2_p6b	Sampling = 8	TYPICAL	2	ALL_ROWS	8	x	50	x
2	6c	c2_p6c	Sampling = 10	TYPICAL	2	ALL_ROWS	10	x	50	x
2	6d	c2_p6d	execute dbms_stats.gather_schema_stats ('HR',	TYPICAL	2	ALL_ROWS	2 (default), debería usar el auto	x	50	x
2	6e	c2_p6e	execute dbms_stats.gather_schema_stats	TYPICAL	2	ALL_ROWS	8, debería usar el auto	x	50	x
2	7a	c2_p7a	opt_mode = choose	TYPICAL	2	CHOOSE	2	x	50	x
2	7b	c2_p7b	opt_mode = choose	TYPICAL	2	CHOOSE	8	x	50	x
2	8a	c2_p8a	fetch = 100	TYPICAL	2	ALL_ROWS	2	x	100	x
2	8b	c2_p8b	fetch = 100	TYPICAL	2	ALL_ROWS	8	x	100	x
2	8c	c2_p8c	fetch = 150	TYPICAL	2	ALL_ROWS	2	x	150	x
2	8d	c2_p8d	fetch = 150	TYPICAL	2	ALL_ROWS	8	x	150	x
2	8e	c2_p8e	fetch = 200	TYPICAL	2	ALL_ROWS	2	x	200	x
2	8f	c2_p8f	fetch = 200	TYPICAL	2	ALL_ROWS	8	x	200	x
2	9a	c2_p9a	Bind: job_title; job_id	TYPICAL	2	ALL_ROWS	2	Job_title --> trabajo; job_id --> trabajo_id	50	x
2	9b	c2_p9b	Bind: job_title; job_id	TYPICAL	2	ALL_ROWS	8	Job_title --> trabajo; job_id --> trabajo_id	50	x
2	10a	c2_p10a	Vista emp_details_view	TYPICAL	2	ALL_ROWS	2	X	50	x
2	10b	c2_p10b	Vista emp_details_view	TYPICAL	2	ALL_ROWS	8	X	50	x
2	10c	c2_p10c	Vista emp_details_view + bind job_title	TYPICAL	2	ALL_ROWS	2	Job_title --> trabajo; job_id --> trabajo_id	50	x
2	10d	c2_p10d	Vista emp_details_view + bind job_title	TYPICAL	2	ALL_ROWS	8	Job_title --> trabajo; job_id --> trabajo_id	50	x
2	11a	c2_p11a	opt_mode = first_rows	TYPICAL	2	FIRST_ROWS	2	x	50	x
2	11b	c2_p11b	opt_mode = first_rows	TYPICAL	2	FIRST_ROWS	8	x	50	x
2	12a	c2_p12a	Unificación de subconsultas	TYPICAL	2	ALL_ROWS	2	x	50	x
2	12b	c2_p12b	Unificación de subconsultas	TYPICAL	2	ALL_ROWS	8	x	50	x

Consulta	ID_mejora	Archivo	Coste (%CPU)	E-Rows	A-Rows	E-Time	A-Time	LRs	PRs	Optimizador
2	0a	c2_p0a; c2_p00	181 (100%)	246	6	0:00:03	00:00:00.14; 00:00:00.26	558	121	ALL_ROWS
2	0b	c2_p0b; c2_p00b	181 (100%)	246	6	0:00:03	00:00:00.08; 00:00:00.06	558	0	ALL_ROWS
2	1	c2_p1	181 (100%)	246	6	0:00:03	00:00:00.08	558	0	ALL_ROWS
2	1b	c2_p1b	181 (100%)	246	6	0:00:03	00:00:00.08	558	0	ALL_ROWS
2	2	c2_p2	181 (100%)	246	6	0:00:03	00:00:00.08	558	0	ALL_ROWS
2	3	c2_p3	181 (100%)	246	6	0:00:03	00:00:00.07	558	0	ALL_ROWS
2	4	c2_p4	181 (100%)	246	6	0:00:03	00:00:00.06	558	0	ALL_ROWS
2	5	c2_p5	181 (100%)	246	6	0:00:03	00:00:00.06	558	0	ALL_ROWS
2	6a	c2_p6a	181 (100%)	281	6	0:00:03	00:00:00.05	558	0	ALL_ROWS
2	6b	c2_p6b	181 (100%)	281	6	0:00:03	00:00:00.07	558	0	ALL_ROWS
2	6c	c2_p6c	181 (100%)	281	6	0:00:03	00:00:00.06	558	0	ALL_ROWS
2	6d	c2_p6d	181 (100%)	246	6	0:00:03	00:00:00.05	558	0	ALL_ROWS
2	6e	c2_p6e	181 (100%)	281	6	0:00:03	00:00:00.06	558	0	ALL_ROWS
2	7a	c2_p7a	181 (100%)	246	6	0:00:03	00:00:00.06	558	0	CHOOSE
2	7b	c2_p7b	181 (100%)	281	6	0:00:03	00:00:00.07	558	0	CHOOSE
2	8a	c2_p8a	181 (100%)	246	6	0:00:03	00:00:00.06	558	0	ALL_ROWS
2	8b	c2_p8b	181 (100%)	281	6	0:00:03	00:00:00.06	558	0	ALL_ROWS
2	8c	c2_p8c	181 (100%)	246	6	0:00:03	00:00:00.05	558	0	ALL_ROWS
2	8d	c2_p8d	181 (100%)	281	6	0:00:03	00:00:00.05	558	0	ALL_ROWS
2	8e	c2_p8e	181 (100%)	246	6	0:00:03	00:00:00.06	558	0	ALL_ROWS
2	8f	c2_p8f	181 (100%)	281	6	0:00:03	00:00:00.06	558	0	ALL_ROWS
2	9a	c2_p9a	175 (100%)	157	6	0:00:03	00:00:00.06	550	0	ALL_ROWS
2	9b	c2_p9b	175 (100%)	281	6	0:00:03	00:00:00.07	550	0	ALL_ROWS
2	10a	c2_p10a	1068 (100%)	18924	6	00:00:13	00:00:00.12	744	0	ALL_ROWS
2	10b	c2_p10b	1445 (100%)	27775	6	0:00:18	00:00:00.12	744	0	ALL_ROWS
2	10c	c2_p10c								
2	10d	c2_p10d								
2	11a	c2_p11a	181 (100%)	246	6	0:00:03	00:00:00.07	558	0	FIRST_ROWS
2	11b	c2_p11b	181 (100%)	281	6	0:00:03	00:00:00.06	558	0	FIRST_ROWS
2	12a	c2_p12a	137 (100%)	246	6	0:00:02	00:00:00.05	426	0	ALL_ROWS
2	12b	c2_p12b	137 (100%)	281	6	0:00:02	00:00:00.04	426	0	ALL_ROWS

Tabla 16: SQL Developer - Resultados Consulta 2

Consulta 3:

Para cada prueba de las descritas en las siguientes tablas adjuntas, se presentan a continuación lo resultados más relevantes:

- *C3_p0b*: La CPU utilizada cuando empezó la llamada se ha reducido a la mitad (de 23 a 11), el tiempo de ejecución también disminuye ('DB time' 77 vs 11); ya no necesita pedir espacio libre en buffer (en 0a tuvo que pedir 121, en 0b 0); ya no hay lecturas físicas, así que no hay peticiones de IO (en 0a fueron 27); el número de llamadas de usuario permanece igual (16).

- *C3_p1*: El uso de la CPU ha disminuido (de 11 a 8), igual que el Dbtime y el tiempo de parse; el resto de parámetros no muestra diferencias remarcables.

- *C3_p2*: Sin diferencia respecto a 0b. Respecto a 1, ha aumentado el uso de la CPU (de 8 a 12), el tiempo Dbtime y el tiempo de parse, lo que la convierte en una opción menos recomendable.

- *C3_p3*: Respecto a 0b no hay grandes diferencias, tan sólo un tiempo ligeramente superior. Respecto a 2, se aprecia una leve mejora en el uso de la CPU, debido a la mejora del nivel de optimización PL/SQL. En cambio la

prueba 1, donde este el nivel del optimizador PL/SQL vale 2 en vez de 1, sigue obteniendo mejores resultados y más diferenciadores.

- *C3_p4*: Vuelve a valores similares a los de la prueba 1, donde el nivel de optimización PL/SQL era 2. De hecho, los tiempos marcados en el execute plan de Autotrace SQL Dev son: 0'452 (plsql = 0), 0'343 (plsql = 1) y 0'234 (plsql = 2 y 3). Entre las pruebas 1 y 4 se aprecia en el segundo un leve descenso del tiempo de parse, pero tan leve que apenas es relevante.

- *C3_p5*: Respecto a la prueba 2, en la que el nivel de estadísticas era 'ALL', el uso de CPU (12 a 10) y el DBtime ha disminuido, igual que el tiempo de Autotrace SQL Dev (0'452 a 0'218) por lo que mantener el nivel en 'TYPICAL' es más recomendable (prueba 2). Respecto a la prueba 0b (donde plsql_opt = 2), los resultados son mejores que en la prueba 5. Conclusión: mejor nivel 'TYPICAL' y plsql = 2.

- *C3_p6a*: Notable incremento del uso de CPU y DBtime (de 11 a 27 en ambos), y más tiempo de parse (4 a 18). Menos recomendable que la configuración de 0b.

- *C3_p6b*: Respecto a 0b, se incrementa el uso de CPU y DBtime (de 11 a 22 en ambos parámetros de v$mystat), se cuadruplica el tiempo de parse (de 4 a 15); por lo que la configuración 0b es más adecuada en este caso. La consulta no es tan elaborada como para que ordenar un tamaño de muestreo más grande merezca la pena; sólo provoca retardos y más accesos.

- *C3_p6c*: Aumenta el uso de CPU de 11 a 27, como el Dbtime, y casi cinco veces más tiempo en parse (4 vs 19), lo que sigue situando como mejor opción a 0b. En este caso no es necesario un muestreo elevado.

- *C3_p6d*: Resultados no concluyentes respecto a 0b. Hay un leve impacto en 6d, pero no el suficiente como para seleccionar 0b mejor que 6d.

- *C3_p6e*: Se vuelve a valores similares a la prueba 6b, e incluso se empeoran (tiempo CPU = 22 vs 26; Dbtime = 22 vs 25; tiempo parse = 15 vs 17). Si se compara con los resultados de la configuración 0b, esta sale beneficiada: tiempo en CPU de 0b = 11 vs 6e = 26; DBtime (0b) = 11 vs (6e) = 25; tiempo parse = 4 vs 17. Conclusión: 0b es mejor ajuste.

- *C3_p7a*: Resultados levemente mejores que en 0b (7a vs 0b): tiempo CPU = 9 vs 11, Dbtime = 10 vs 11, tiempo parse = 3 vs 4. Sin embargo son muy leves, por lo que no llegan a ser concluyentes. 7a conlleva una ordenación extra en memoria (89 vs 88 de 0b), pero reordena más líneas que 0b, y en el mismo número de llamadas de usuario (16).

- *C3_p7b*: 6b, la configuración equivalente en modo 'ALL_ROWS', resulta ser ligeramente mejor que 7b: (6b vs 7b) tiempo CPU = 22 vs 26, DBtime = 22 vs 27, tiempo parse = 15 vs 19. /b conlleva un reordenación extra (89 frente a las

88 de 6b), igual que ocurría en 7a. Mismo número de llamadas de usuario => en este caso concreto, sería mejor el optimizador en modo 'ALL_ROWS'. Si se compara con 0b, que es el punto de partida, la CPU usada es mucho menor en 0b (11 vs 26), lo mismo que para DBtime y tiempo de parse (4 vs 19). Por lo tanto, entre 0b y 7b, es mejor 0b, con el modo optimizador = 'ALL_ROWS' y un tamaño de muestreo = 2 en vez de 8.

- *C3_p8a*: En contraposición con 0b, 8a exige más recursos: tiempo CPUy Dbtime = 11 vs 13, tiempo parse = 4 vs 6. Sin embargo, decrece el número de llamadas de usuario porque en cada una de ellas extrae más información que 0b. Las diferencias no son lo suficientemente grandes como para considerar 0b claramente superior a 8a, por lo que ambos resultados se consideran satisfactorios en este caso.

- *C3_p8b*: Para un mismo valor de fetch, hay diferencias entre las configuraciones 8a y 8b, achacables a la diferencia en el tamaño de muestreo (8a < 8b): tiempo CPU = 11 vs 22, Dbtime = 13 vs 21, tiempo parse = 6 vs 16. 8b requiere más llamadas recursivas. Por tanto, ante igual fetch en una base de datos con estas características y esta consulta, mejor utilizar un tamaño de muestreo inferior.

- *C3_p8c*: Respecto a 8a, 8c conlleva menos uso de CPU y Dbtime (13 vs 9), menos tiempo de parse (6 vs 3) y menor número de llamadas de usuario (10 vs 8). La mejora puede estar debida a esta reducción del número de llamadas. Sin embargo, las mejoras no son lo suficientemente visibles como para seleccionar claramente una de las opciones antes que la otra. Si se compara con 0b, también se aprecia una leve mejora, posiblemente debida al aumento de información extraída unitariamente (nivel de fetch).

- *C3_p8d*: Para un valor fetch similar (8), la configuración 8b ha resultado dar mejores datos, aunque tan sólo ligeramente, por lo que no se descartaría completamente la 8d. Esta mejora puede deberse a un mejor ajuste del parámetro fetch a la consulta 3 en esta base de datos concreta, lo que muestra la importancia de adecuar los niveles de los parámetros a cada caso concreto. En cambio, si se compara con la configuración 0b, esta resulta claramente mejor: sus valores de tiempo de uso de CPU, DBtime y tiempo de parse son entre la mitad y un cuarto de lo que valen en 8d, a pesar de que 8d sólo emplea 8 user calls frente a las 16 de 0b.

- *C3_p8e*: El aumento del valor fetch no supone mejora perceptible en el rendimiento, los valores de tiempo de uso de CPU, DBtime y tiempo de parse son muy similares. Esto unido a la recomendación de Oracle de procurar no utilizar valores fetch muy altos señalan a 8c como preferible a priori. Respecto a 0b, los resultados en los parámetros indicados son similares, incluso favorables para 8e, a lo que se le añade que este emplea la mitad de llamadas de usuario que 0b.

- *C3_p8f*: Al igual que en 8e, el aumento del valor fetch respecto a su homólogo con fetch = 150, 8d, no ha supuesto una visible mejora, por lo que será preferible 8d al cumplir las recomendaciones de Oracle. Comparando con 0b, esta es claramente preferible a 8f, dado que consume menos recursos y tarda menos tiempo. Conclusión: aumentar el fetch no siempre es la solución, depende de cada caso y bd concretos.

- *C3_p9a*: No hay cambios respecto a 0b en los parámetros de tiempo de CPU y Dbtime, pero en 9a, no hay tiempo de parse en CPU aunque tanto en 0b como en 9a se detecta un hard parse (v$mystat). Por la reducción de esta operación, se escogería 9a antes que 0b.

- *C3_p9b*: Entre 9a y 9b, 9a es preferible: consume menos recursos (tiempo en CPU = 11 vs 25, Dbtime = 12 vs 25, número de ejecuciones necesarias = 2 vs 7, número de parses, y menos llamadas recursivas).

- *C3_p10a*: 10a mejora el rendimiento de la consulta respecto a 0b: (0b vs 10a) uso CPU = 11 vs 9, Dbtime = 11 vs 10, parse time cpu = 4 vs 3.

- *C3_p10b*: 10a es claramente preferible a 10b, ya que 10b duplica el consumo de recursos. Tiene sentido ya que la tabla involucrada contiene muchos datos y el aumento respectivo del tamaño de muestreo requiere más recursos.

- *C3_p11a*: El valor de CPU usada es el mismo entre 0b y 11a (11), pero DBtime favorece a 0b (11 vs 13), y el parse time a 11a debido a la presencia de variables de enlace (4 vs 0). Así pues, los resultados no son concluyentes, y debería probarse esta combinación en una base de datos más extensa.

- *C3_p11b*: Entre 0b y 11b, 0b sería la preferida. 0b consume menos CPU (11 vs 20) y DBtime (11 vs 22), aunque sí tiene tiempo invertido en parses (0b no tiene binds).

- *C3_p12a* y *c3_p12b*: Opciones descartadas al 100%, cuando se usa la forma original de la consulta.

- *C3_p12c*: La combinación de el modo del optimizador 'FIRST_ROWS' y la vista emp_details_view mejora el resultado de 12b, pero no lo suficiente como para ser tenido en cuenta, por lo que al igual que 12a, queda descartado como alternativa optimizada de la consulta original.

Tabla 17: SQL Developer - Descripción pruebas Consulta 3

Consulta	ID_mejora	Archivo	Descripción	Statistics_level	plsql_optimize_level (SQL Dev GUI)	Optimizer_mode (cmd / SQL dev)	Opt_dynamic_sampling (v$parameters)	Bind variable	Fetch rows (SQL Dev GUI)	Result_cache_mode
3	0a	c3_p0a	Ejecución inicial	TYPICAL	2	ALL_ROWS	2	X	50	x
3	0b	c3_p0b	Reejecución	TYPICAL	2	ALL_ROWS	2	x	50	x
3	1	c3_p1	alter session set statistics_level='ALL'	ALL	2	ALL_ROWS	2	x	50	x
3	2	c3_p2	plsql_opt_level = 0 (GUI)	ALL	0	ALL_ROWS	2	x	50	x
3	3	c3_p3	plsql_opt_level = 1 (GUI)	ALL	1	ALL_ROWS	2	x	50	x
3	4	c3_p4	plsql_opt_level = 3 (GUI)	ALL	3	ALL_ROWS	2	x	50	x
3	5	c3_p5	0b + 2	TYPICAL	0	ALL_ROWS	2	x	50	x
3	6a	c3_p6a	Sampling = 5	TYPICAL	2	ALL_ROWS	5	x	50	x
3	6b	c3_p6b	Sampling = 8	TYPICAL	2	ALL_ROWS	8	x	50	x
3	6c	c3_p6c	Sampling = 10	TYPICAL	2	ALL_ROWS	10	x	50	x
3	6d	c3_p6d	execute dbms_stats.gather_schema_stats ('HR', dbms_stats.auto_sample_size);	TYPICAL	2	ALL_ROWS	2 (default), debería usar el auto	x	50	x
3	6e	c3_p6e	execute dbms_stats.gather_schema_stats ('HR', dbms_stats.auto_sample_size);	TYPICAL	2	ALL_ROWS	8, debería usar el auto	x	50	x
3	7a	c3_p7a	opt_mode = choose	TYPICAL	2	CHOOSE	2	x	50	x
3	7b	c3_p7b	opt_mode = choose	TYPICAL	2	CHOOSE	8	x	50	x
3	8a	c3_p8a	fetch = 100	TYPICAL	2	ALL_ROWS	2	x	100	x
3	8b	c3_p8b	fetch = 100	TYPICAL	2	ALL_ROWS	8	x	100	x
3	8c	c3_p8c	fetch = 150	TYPICAL	2	ALL_ROWS	2	x	150	x
3	8d	c3_p8d	fetch = 150	TYPICAL	2	ALL_ROWS	8	x	150	x
3	8e	c3_p8e	fetch = 200	TYPICAL	2	ALL_ROWS	2	x	200	x
3	8f	c3_p8f	fetch = 200	TYPICAL	2	ALL_ROWS	8	x	200	x
3	9a	c3_p9a	Bind: elapsed_years	TYPICAL	2	ALL_ROWS	2	Nº de años atrás que cuento	50	x
3	9b	c3_p9b	Bind: elapsed_years	TYPICAL	2	ALL_ROWS	8	Nº de años atrás que cuento	50	x
3	10a	c3_p10a	Vista emp_details_view	TYPICAL	2	ALL_ROWS	2	X	50	x
3	10b	c3_p10b	Vista emp_details_view	TYPICAL	2	ALL_ROWS	8	X	50	x
3	11a	c3_p11a	Bind: elapsed_years + emp_details_view	TYPICAL	2	ALL_ROWS	2	Nº de años atrás que cuento	50	x
3	11b	c3_p11b	Bind: elapsed_years + emp_details_view	TYPICAL	2	ALL_ROWS	8	Nº de años atrás que cuento	50	x
3	12a	c3_p12a	opt_mode = first_rows	TYPICAL	2	FIRST_ROWS	2	x	50	x
3	12b	c3_p12b	opt_mode = first_rows	TYPICAL	2	FIRST_ROWS	8	x	50	x
3	12c	c3_p12c	opt_mode = first_rows + emp_details_view	TYPICAL	2	FIRST_ROWS	2	X	50	x

Consulta	ID_mejora	Archivo	Coste (%CPU)	E-Rows	A-Rows	E-Time	A-Time	LRs	PRs	Optimizador
3	0a	c3_p0a	50 (100%)	1	10	0:00:01	00:00:00.23	243	108	ALL_ROWS
3	0b	c3_p0b	50 (100%)	1	10	0:00:01	00:00:00.05	243	0	ALL_ROWS
3	1	c3_p1	50 (100%)	1	10	0:00:01	00:00:00.03	243	0	ALL_ROWS
3	2	c3_p2	50 (100%)	1	10	0:00:01	00:00:00.05	243	0	ALL_ROWS
3	3	c3_p3	50 (100%)	1	10	0:00:01	00:00:00.05	243	0	ALL_ROWS
3	4	c3_p4	50 (100%)	1	10	0:00:01	00:00:00.03	243	0	ALL_ROWS
3	5	c3_p5	50 (100%)	1	10	0:00:01	00:00:00.04	243	0	ALL_ROWS
3	6a	c3_p6a	96 (100%)	1	10	0:00:02	00:00:00.05	243	0	ALL_ROWS
3	6b	c3_p6b	96 (100%)	1	10	0:00:02	00:00:00.04	243	0	ALL_ROWS
3	6c	c3_p6c	96 (100%)	1	10	0:00:02	00:00:00.05	243	0	ALL_ROWS
3	6d	c3_p6d	50 (100%)	1	10	0:00:01	00:00:00.06	243	0	ALL_ROWS
3	6e	c3_p6e	96 (100%)	1	10	0:00:02	00:00:00.05	243	0	ALL_ROWS
3	7a	c3_p7a	50 (100%)	1	10	0:00:01	00:00:00.04	243	0	CHOOSE
3	7b	c3_p7b	96 (100%)	1	10	0:00:02	00:00:00.05	243	0	CHOOSE
3	8a	c3_p8a	50 (100%)	1	10	0:00:01	00:00:00.05	243	0	ALL_ROWS
3	8b	c3_p8b	96 (100%)	1	10	0:00:02	00:00:00.04	243	0	ALL_ROWS
3	8c	c3_p8c	50 (100%)	1	10	0:00:01	00:00:00.04	243	0	ALL_ROWS
3	8d	c3_p8d	96 (100%)	1	10	0:00:02	00:00:00.05	243	0	ALL_ROWS
3	8e	c3_p8e	50 (100%)	1	10	0:00:01	00:00:00.04	243	0	ALL_ROWS
3	8f	c3_p8f	96 (100%)	1	10	0:00:02	00:00:00.05	243	0	ALL_ROWS
3	9a	c3_p9a	50 (100%)	1	10	0:00:01	00:00:00.05	243	0	ALL_ROWS
3	9b	c3_p9b	96 (100%)	1	10	0:00:02	00:00:00.05	243	0	ALL_ROWS
3	10a	c3_p10a	50 (100%)	1	10	0:00:01	00:00:00.04	243	0	ALL_ROWS
3	10b	c3_p10b	96 (100%)	1	10	0:00:02	00:00:00.05	243	0	ALL_ROWS
3	11a	c3_p11a	50 (100%)	1	10	0:00:01	00:00:00.05	243	0	ALL_ROWS
3	11b	c3_p11b	96 (100%)	1	10	0:00:02	00:00:00.05	243	0	ALL_ROWS
3	12a	c3_p12a	1776 (100%)	4744	10	0:00:22	00:00:00.04	243	0	FIRST_ROWS
3	12b	c3_p12b	5805 (100%)	41506	10	0:01:10	00:00:00.05	265	0	FIRST_ROWS
3	12c	c3_p12c	1776 (100%)	4744	10	0:00:22	00:00:00.03	243	0	FIRST_ROWS

Tabla 18: SQL Developer - Resultados Consulta 3

5.3 Optimización de consultas en TOAD

Con el fin de mantener la coherencia de criterio de evaluación entre las pruebas y sus potenciales mejoras, los parámetros empleados en TOAD son los mismos a los de SQL Developer, a excepción del tiempo estimado. Es decir, serán: coste, tiempo real de ejecución, líneas estimadas (E-rows), líneas reales obtenidas (A-rows), número de lecturas lógicas (LRs), número de lecturas físicas (PRs) y modo del optimizador empleado.

En este caso las fuentes de información para dichos parámetros de evaluación están concentradas en el navegador de tracefile (‘Trace File Browser’):

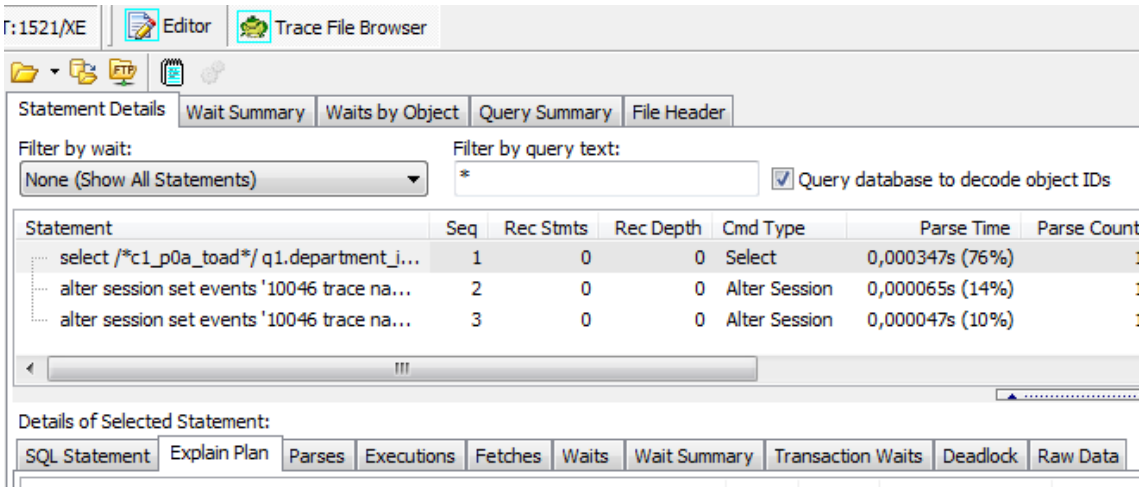


Ilustración 31: TOAD - Tracefile Browser

Con el fin de diferenciar las pruebas de las ejecutadas en SQL Developer, para Toad se seguirá la siguiente nomenclatura: "cX_pY_toad", siendo X el número de la consulta (1, 2 o 3) e Y el número identificador de la prueba dentro de una consulta. En los resultados hay que tener en cuenta que el número de escrituras siempre será 0, ya que todas las sentencias de los casos de prueba son de tipo 'select'. Las alternativas mencionadas en esta sección pueden ser consultadas en los anexos 7.3.

Consulta 1:

Las primeras pruebas, *c1_p0a_toad* y *c1_p0b_toad*, se incluyen en el estudio para generar un punto de partida frente al cual valorar en qué medida se ha mejorado el rendimiento en cada caso. La primera incluye lecturas tanto lógicas como físicas debido a que no existe información previa que el sistema pueda consultar para ahorrar tiempo y accesos, mientras que esto sí se da en la segunda y por ello sólo requiere lecturas lógicas.

Los casos *c1_p1_toad* y *c1_p2_toad* experimentan con las opciones de refactorizado siguiendo los métodos Oracle y ANSI. En estos se sustituyen respectivamente las operaciones join - on por from - where y from - inner join. En ambos casos los resultados son similares, y no se aprecian mejoras significativas.

En *c1_p3_toad*, se utiliza la opción "*Auto Optimize SQL*" y "*Advanced Query Optimization*" de las opciones para optimizar. Esto conduce a otro módulo de TOAD llamado "Dell SQL Optimizer for Oracle 9.2.0", donde se realizan los procesos automáticos de optimización. En el anexo 7.3 se han incluido imágenes de los diálogos que conducen a este módulo.

Toad tiene tres métodos distintos de optimizar una consulta: 1) SQL Query rewrite, 2) by plan control, y 3) por batch. Se ha escogido el primero, y el entorno se ha definido como variable en cuanto a tipo de aplicación en la que se ejecuta, si la consulta

es estática o dinámica y si existen picos de trabajo (para más detalles, consultar el anexo).

Tras ejecutar una vez la consulta que se quiere optimizar, se ha seguido el procedimiento de auto-optimización, lo que ha resultado en 130 escenarios alternativos, con diferentes valores de costes de CPU, tiempos de ejecución y otros parámetros. Teniendo en cuenta que se han respetado los niveles por defecto de la 'inteligencia' del optimizador y la generación de índices (2), destacan las siguientes mejoras:

- Por mejora en tiempo de ejecución: la alternativa 29.

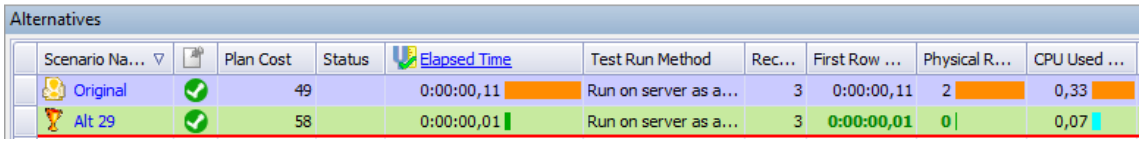


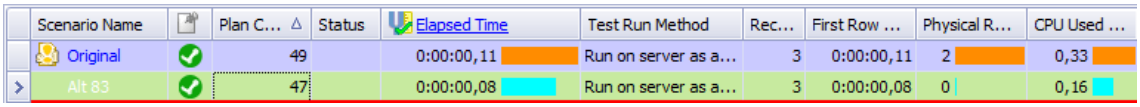
Ilustración 32: TOAD - Muestra comparación de alternativas (I)

```
select /*+ USE_HASH(Q3,Q2) ORDERED */ q1.department_id,
       q1.department_name,
       q1.region_name,
       q3.manager_id,
       q3.subordinados
  FROM (select DEPARTMENTS1.department_id,
               DEPARTMENTS1.department_name,
               regions.region_name
          FROM hr.departments DEPARTMENTS1
               INNER JOIN hr.locations
                  ON DEPARTMENTS1.location_id = locations.location_id
               INNER JOIN hr.countries
                  ON locations.country_id = countries.country_id
               INNER JOIN hr.regions
                  ON countries.region_id = regions.region_id) q1
       INNER JOIN (select distinct DEPARTMENTS2.department_id,
                                   DEPARTMENTS2.manager_id
                     FROM hr.departments DEPARTMENTS2
                          INNER JOIN hr.employees EMPLOYEES1
                             ON DEPARTMENTS2.manager_id =
EMPLOYEES1.employee_id
                          INNER JOIN hr.jobs
                             ON EMPLOYEES1.job_id = jobs.job_id
                    WHERE jobs.job_title like '%Finance%') q2
          ON q1.department_id = q2.department_id
       INNER JOIN (select EMPLOYEES2.manager_id,
                          count(*) as subordinados
                     FROM hr.departments DEPARTMENTS3
                          INNER JOIN hr.employees EMPLOYEES2
                             ON DEPARTMENTS3.manager_id =
EMPLOYEES2.manager_id
                    WHERE EMPLOYEES2.hire_date > '01/01/2000'
                    group by EMPLOYEES2.manager_id) q3
          ON q2.manager_id = q3.manager_id
 order by q3.manager_id asc;
```

El plan asociado a esta mejora requiere 26 pasos, frente a los 30 de la original. La mejora se debe en gran medida al cambio de los bucles anidados del plan original a hash joins, obligado por el hint "/*+ USE_HASH(Q3,Q2) ORDERED */". Con ello se

consiguen reducir los tiempos de ejecución (0:00:00,11 a 0:00:00,01) y conseguir una mejora del 45'45%, pero a cambio son necesarias más escaneados de tabla que dan lugar a un mayor coste de CPU (pasa de 49 a 58).

- Por mejora en coste: la alternativa 83.



Scenario Name		Plan C...	Status	Elapsed Time	Test Run Method	Rec...	First Row ...	Physical R...	CPU Used ...
Original	✓	49		0:00:00,11	Run on server as a...	3	0:00:00,11	2	0,33
Alt 83	✓	47		0:00:00,08	Run on server as a...	3	0:00:00,08	0	0,16

Ilustración 33: TOAD - Muestra comparación de alternativas (II)

```
select q1.department_id,
       q1.department_name,
       q1.region_name,
       q3.manager_id,
       q3.subordinados
  FROM (select DEPARTMENTS1.department_id,
               DEPARTMENTS1.department_name,
               regions.region_name
          FROM hr.departments DEPARTMENTS1
               INNER JOIN hr.regions
                  ON 1 = 1
               INNER JOIN hr.locations
                  ON DEPARTMENTS1.location_id = locations.location_id
+ 0
               INNER JOIN hr.countries
                  ON regions.region_id = countries.region_id + 0
                     AND locations.country_id = countries.country_id
|| '') q1
       INNER JOIN (select /*+ USE_NL(JOBS) */ distinct
DEPARTMENTS2.department_id,
                          DEPARTMENTS2.manager_id
                     FROM hr.departments DEPARTMENTS2
                          INNER JOIN hr.employees EMPLOYEES1
                             ON DEPARTMENTS2.manager_id =
EMPLOYEES1.employee_id
                          INNER JOIN hr.jobs
                             ON EMPLOYEES1.job_id = jobs.job_id
                    WHERE jobs.job_title like '%Finance%') q2
          ON q1.department_id = q2.department_id
       INNER JOIN (select EMPLOYEES2.manager_id,
                          count(*) as subordinados
                     FROM hr.departments DEPARTMENTS3
                          INNER JOIN hr.employees EMPLOYEES2
                             ON DEPARTMENTS3.manager_id =
EMPLOYEES2.manager_id
                    WHERE EMPLOYEES2.hire_date > '01/01/2000'
                    group by EMPLOYEES2.manager_id) q3
          ON q2.manager_id = q3.manager_id
 order by q3.manager_id asc;
```

En este caso la mejora se ha centrado en el coste de la ejecución, que ha pasado de 49 a 47 a pesar de contar con 33 pasos en el plan de ejecución. También se ha reducido el tiempo, de 0:00:00,11 a 0:00:00,08 (mejora un 27'27%). Aunque otros parámetros también han sido reducidos, no hay diferencias significativas.

Para la prueba *c1_p4_toad*, se ha planificado emplear el optimizador de Oracle. Sin embargo, al ser una funcionalidad reservada al pago de una licencia, no ha sido posible: "ORA-13717: Tuning Package License is needed for using this feature.". Otros fallos relacionados con privilegios de acceso a otros paquetes también privados como SQL_Tune han aparecido ("ORA-06512: at "SYS.DBMS_SQLTUNE", line 625").

En *c1_p6_toad*, se ha utilizado únicamente la función "Auto Optimize SQL". El resultado es un conjunto de 129 alternativas seleccionadas sobre un total de 821 generadas, ordenadas según uno de los criterios de rendimiento. El tiempo transcurrido es el aplicado por defecto.

- Criterio 1 (*c1_p6a_toad*): tiempo transcurrido. La ejecución de la sentencia original ha necesitado 0'01 segundos, y en función de este criterio no existen alternativas mejores pero sí similares (con el mismo valor): 79, 100, 120 y 121. Introduciendo un criterio secundario, el coste, se discriminarían estas alternativas: C (original) = 49; C (79) = 363; C (100) = 280; C (120) = 280; C (121) = 116. Por lo tanto, bajo estas condiciones AO SQL no ha encontrado una alternativa mejor.

- Criterio 2 (*c1_p6b_toad*): coste de la ejecución. La sentencia original conlleva un coste de 49, y el optimizador ha encontrado una alternativa que lo reduce a 47 (alternativa 83, ver anexos 7.3). Sin embargo, esta mejora se obtiene en detrimento de otros parámetros como el nivel de uso de CPU, que se eleva de 11 (original) a 155.

Las pruebas *c1_p5_toad*, *c1_p7_toad* y *c1_p8_toad* giran en torno a los niveles de 'inteligencia' del optimizador y la generación de índices para la sesión actual. Cuanto mayor es el nivel, más escenarios alternativos considera al buscar una alternativa mejor. En el caso del optimizador, se incrementa progresivamente el número de hints que tiene en cuenta al generar nuevas posibilidades. Para los índices, un nivel superior supone generar más sets de índices. Ambas decisiones afectarán al tiempo de análisis del Dell SQL Optimizer, de manera que cuanto mayor sea el nivel más tardarán en generarse las alternativas, llegando al orden de horas.

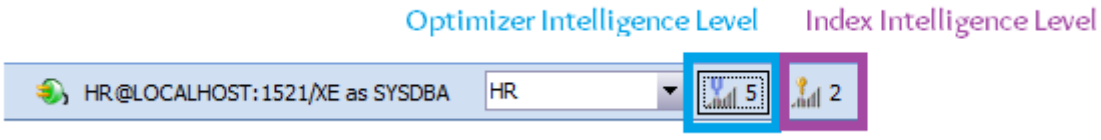


Ilustración 34: TOAD - Intelligence Levels

En las anteriores ejecuciones se mantuvieron los niveles por defecto, 2. En las pruebas 5, 7 y 8 se modifican: Tabla 11: TOAD - Session Settings: OptimizerIL e IndexIL.

c1_p5_toad aumenta el nivel del optimizador a su máximo (5) y siguiendo los dos criterios de tiempo transcurrido y coste del plan se han obtenido las siguientes mejoras:

- Criterio 1 (*c1_p5a_toad*): tiempo transcurrido. De manera similar al caso de la prueba c1_p3_toad, la alternativa con mejor marca de tiempo también conlleva un mayor uso de los recursos y por tanto de coste. En este caso el tiempo se reduce a 0'01 segundos pero aumenta el coste hasta 60 y genera más operaciones de lectura lógica. Alternativa 185.

- Criterio 2 (*c1_p5b_toad*): coste de ejecución. La mejora en el coste es leve, pasando de 49 a 47, pero a cambio aumenta en gran medida el tiempo transcurrido (0'08 segundos). Alternativa 149.

En la prueba *c1_p7_toad*, se invierten los niveles, y en este caso es el nivel de los índices el que llega a su máximo. Partiendo de un total de 130 escenarios generados:

- Criterio 1 (*c1_p7a_toad*): tiempo transcurrido. No se han generado alternativas que mejoren el tiempo de la consulta original, que en esta ejecución es igual a 0'01 segundos. Las alternativas más cercanas ascienden a 0'02 segundos, pero de costes distintos: C (12) = 279; C (14) = 174; C (29) = 58; C (55) = 80; C (62) = 78; C (80) = 208; C (90) = 144; C (92) = 143; C (100) = 280. Esto dejaría a la alternativa 29 como la siguiente mejor en términos de tiempo y coste, pero no es mejor que la original.

- Criterio 2 (*c1_p7b_toad*): coste de ejecución. Según este criterio, sí existe una alternativa que mejora el valor original (49) y lo disminuye hasta 47: alternativa 83. El tiempo en cambio aumenta de 0'01 a 0'05 segundos. Esta alternativa también requiere más reordenaciones de líneas, más consumo de CPU y más lecturas lógicas durante la sesión.

Por último, c1_p8_toad fuerza ambos niveles al máximo, 5. El tiempo transcurrido para generar los escenarios y probarlos antes de incluirlos en el subgrupo 'alternativas' ha sido superior al de las anteriores pruebas con creces, superando las 5h. Finalmente se generan 502 escenarios posibles.

- Criterio 1 (*c1_p8a_toad*): tiempo transcurrido. Teniendo en cuenta que la consulta original emplea 0'01 segundos, no hay escenario alguno en el que se mejore o se iguale este resultado. Las siguientes mejores alternativas tardan 0'02 segundos, y atendiendo a sus costes, puede escogerse la mejor: C (37) = 279; C (45) = 60; C (147) = 57; C (153) = 363; C (181) = 62; C (185) = 60; C (299) = 80; C (355) = 371; C (364) = 144; C (406) = 280; C (431) = 66; C (469) = 60; C (474) = 136; C (490) = 292. Por lo tanto, la siguiente alternativa a la original sería la 147, con un coste de 57 (superior al 49 de la original).

- Criterio 2 (*c1_p8b_toad*): coste de ejecución. Según este criterio, se ha encontrado una alternativa cuyo coste es inferior a l original (49): la alternativa 149. Su coste es de 47, pero al igual que sucedía en anteriores pruebas con el mismo criterio, esta emplea más tiempo (0'08 segundos) y más recursos (el uso de la CPU aumenta de los 0'08 originales a 0,16).

Como resumen de lo expuesto en relación a esta primera consulta se adjuntan las siguientes tablas, similares a las presentadas anteriormente en SQL Developer:

ID_mejora	Descripción	Detalles
0a_toad	Ejecución inicial	Ejecución de partida, consulta sin modificar. Se esperan lecturas físicas.
0b_toad	Reejecución (se espera mejor resultado)	Segunda ejecución de la consulta original, será el punto de referencia. Objetivo: eliminar la mejora aparente debida exclusivamente a que los datos ya se encuentran en memoria y por tanto no son necesarias lecturas físicas.
1_toad	Refactor > Convert to Oracle Join syntax	Sustituye los 'join - on' por 'from - where'
2_toad	Refactor > Convert to ANSI Join syntax	Sustituye los 'join - on' por 'from - inner join'
3a_toad	Optimize > Auto Optimize SQL + Advances Query Opt.	OptimIntelligence=2; IndexIntelligence = 2 (valores por defecto). De los 130 escenarios alternativos detectados, hay uno que es el mejor en tiempo de ejecución
3b_toad	Optimize > Auto Optimize SQL + Advances Query Opt.	OptimIntelligence=2; IndexIntelligence = 2 (valores por defecto). De los 130 escenarios alternativos detectados, hay uno que es el mejor en uso de CPU
4_toad	Optimize > OEM	No es posible. Funcionalidad exclusiva bajo licencia Oracle adecuada.
5a_toad	Optimize > Auto Optimize SQL + Advances Query Opt.; Session Settings	Optimizer Intelligence pasa de 2 a 5 (el máximo); Index Intelligence permanece en 2. Alternativa con mejor tiempo.
5b_toad	Optimize > Auto Optimize SQL + Advances Query Opt.; Session Settings	Optimizer Intelligence pasa de 2 a 5 (el máximo); Index Intelligence permanece en 2. Alternativa con menor coste.
6a_toad	Optimize > Auto Optimize SQL	Criterio: tiempo.
6b_toad	Optimize > Auto Optimize SQL	Criterio: coste de ejecución.
7a_toad	Optimize > Auto Optimize SQL + Advances Query Opt.; Session Settings	Optimizer Intelligence permanece en 2; Index Intelligence pasa de 2 a 5 (el máximo). Criterio: tiempo.
7b_toad	Optimize > Auto Optimize SQL + Advances Query Opt.; Session Settings	Optimizer Intelligence permanece en 2; Index Intelligence pasa de 2 a 5 (el máximo). Criterio: coste de ejecución.
8a_toad	Optimize > Auto Optimize SQL + Advances Query Opt.; Session Settings	Optimizer Intelligence = 5; Index Intelligence = 5 (ambos al máximo). Criterio: tiempo.
8b_toad	Optimize > Auto Optimize SQL + Advances Query Opt.; Session Settings	Optimizer Intelligence = 5; Index Intelligence = 5 (ambos al máximo). Criterio: coste de ejecución.

Tabla 19: TOAD - Descripción pruebas Consulta 1

Consulta	ID_mejora	Coste	E-Rows	A-Rows	E-Time	A-Time	LRs	PRs	Optimizador
1	0a_toad	49	1	3	NS/NC	0,01205	123	85	All Rows
1	0b_toad	49	1	3	NS/NC	0,141922	123	0	All Rows
1	1_toad	49	1	3	NS/NC	0,078326	123	0	All Rows
1	2_toad	49	1	3	NS/NC	0,099753	123	0	All Rows
1	3a_toad	58	4	3	NS/NC	0,01 seg	160	0	All Rows
1	3b_toad	47	1	3	NS/NC	0,08 seg	124	0	All Rows
1	4_toad								
1	5a_toad	60	4	3	NS/NC	0,01 seg	157	0	All Rows
1	5b_toad	47	1	3	NS/NC	0,08 seg	124	0	All Rows
1	6a_toad	49	1	3	NS/NC	0,01 seg	132	0	All Rows
1	6b_toad	47	1	3	NS/NC	0,06 seg	131	0	All Rows
1	7a_toad	58	4	3	NS/NC	0,02 seg	123	0	All Rows
1	7b_toad	47	1	3	NS/NC	0,05 seg	123	0	All Rows
1	8a_toad	57	1	3	NS/NC	0,02 seg	121	0	All Rows
1	8b_toad	47	1	3	NS/NC	0,08 seg	124	0	All Rows

Tabla 20: TOAD - Resultados Consulta 1

Para esta primera consulta, se han descubierto a lo largo de la batería de pruebas una serie de patrones que merecen especial atención, ya que pueden ser empleados en el futuro para diseñar desde un primer momento consultas más óptimas:

- El hint '/*+ USE_HASH(Q3,Q2) ORDERED */' ha sido el más repetido en aquellas pruebas que tenían como fin encontrar la alternativa que minimizara el tiempo de ejecución. Por ejemplo, es empleado tanto en c1_p3a_toad como en c1_p5a_toad y c1_p7a_toad, y analizando estas tres alternativas en conjunto, se puede observar que c1_p3a_toad y c1_p7a_toad son idénticas entre sí y muy similares a c1_p5a_toad.

En este caso la mejora se debe a forzar un join ordenado de tipo hash entre los resultados de las subconsultas q3 y q2, donde los resultados de q3 son utilizados para encontrar los de q2. Los hash join son el método de join aplicado entre grandes conjuntos de datos, que se unen mediante la construcción de una tabla hash sobre el conjutno más pequeño en base a la join key, y utiliza esta para escanear el conjunto más grande de resultados. La cláusula 'ordered' obliga al optimizador a ejecutar el join entre tablas respetando el orden en que aparecen en la cláusula 'from'. En este caso concreto, se utilizará el subgrupo de empleados que fueron contratados a partir del año 2.000 para encontrar a aquellos que además ocupan un cargo financiero.

- Cuando el objetivo es en cambio minimizar los costes, los hint más frecuentemente aplicados son '/*+ USE_NL(JOBS)*/' (en c1_p3b_toad, c1_p6b_toad, y c1_p7b_toad) y '/*+ LEADING(JOBS) */' (en c1_p5b_toad y c1_p8b_toad). En este caso, las alternativas c1_p3b_toad, c1_p6b_toad y c1_p7b_toad son realmente la misma alternativa, que ha sido hallada en distintas configuraciones de los niveles 'Optimizer Intelligence Level' e 'Index Generator Intelligence Level' donde sin embargo el valor del primero es siempre 2. Entre c1_p5b_toad y c1_p8b_toad sucede lo mismo, son idénticas entre sí, y ambas configuraciones tienen en común que el valor de 'Optimizer Intelligence Level' es 5. De este comportamiento se deduce que en este caso la optimización es independiente del valor que tenga 'Index Generator Intelligence Level'.

  En el caso del hint '/*+ USE_NL(JOBS)*/', este fuerza a la tabla 'jobs' a ser la inner table de la operación join usando bucles anidados ('nested loops'), de manera que por cada ocurrencia de la tabla externa ('outer table') se accederá a 'jobs'. Esto supondrá menor coste ya que en 'departments' (la tabla externa) hay un máximo de 62 valores diferentes para 'department_id', mientras que en 'jobs' hay 105 'job_id' distintos, y si se ejecutara la operación join a la inversa se requerirían más accesos y por lo tanto más consumo.

  El hint '/*+ LEADING(JOBS) */' es muy similar a 'ordered', pero es más versátil y Oracle lo recomienda antes que 'ordered'.

Consulta 2:

Los casos *c2_p1_toad* y *c2_p2_toad* están dedicados a la mejora mediente refactorización de la consulta original siguiendo los estándares Oracle y ANSI respectivamente en la sintaxis join. Al utilizar el refactorizado Oracle, aparece una recomendación de Toad (Regla 5807) según la cual se recomienda evitar las consultas cartesianas y en su lugar emplear el estándar ANSI. Sin embargo, ambos cambios resultan fallidos, ya que si se ejecuta la consulta resultado, los datos obtenidos por estas versiones difieren completamente de los esperados. Se han incluido las versiones refactorizadas en los anexos para ser utilizadas como ejemplo erróneo.

La prueba *c2_p3_toad* genera 192 escenarios, de los cuales sólo 142 se consideran alternativas válidas, en 2h. En este caso se han mantenido los niveles por defecto de Optimizer Intelligence e Index Generation Intelligence, 2. Ambos devuelven los resultados esperados.

- Criterio 1 (*c2_p3a_toad*): tiempo transcurrido. Atendiendo a este criterio, de las 142 alternativas seleccionadas la mejor es la 186, que ejecuta la consulta en 0,06 segundos (frente a los 0,3 de la original) e introduce una mejora del 80%. El uso de CPU también es menor, 0,13 frente a los 0,64 originales, y aunque realiza más operaciones de ordenación ('sorts (rows)'), los tiempos dedicados a parse son menores.

- Criterio 2 (*c2_p3b_toad*): coste de ejecución. El coste original, 181, desciende a 56 con la alternativa 120. Esta además se ejecuta en un tiempo inferior también, 0,15 segundos (lo mejora un 50%), y consume menos recursos de CPU (0,27 en vez de 0,64). Esto es debido a una reducción en el número de escaneos de tablas y un aumento en la cantidad de datos extraída de ellas (table fetch rows).

Al igual que en la consulta 1, *c2_p4_toad* no puede ser llevada a cabo. El método de optimización OEM necesita el paquete Oracle Tuning, que sólo es accesible bajo licencia.

El grupo de pruebas *c2_p6a_toad* se ha llevado a cabo empleando la función Auto Optimize SQL. En el transcurso de 25 minutos, considera 842 escenarios y tras una criba selecciona como alternativas viables a 192.

- Criterio 1 (*c2_p6a_toad*): tiempo transcurrido. La mejor alternativa seleccionada por el optimizador es la 10, que mejora el tiempo de ejecución un 67% al tardar 0,06 segundos frente a los 0,18 originales y con el mismo coste de 181. Sin embargo, existen otras alternativas que también han tardado 0,06 segundos, cuyos costes en cambio difieren: C (10) = 181, C (14) = 181; C (19) = 181; C (41) = 185; C (43) = 186; C (72) = 182; C (82) = 3044; C (102) = 186; C (159) = 186; C (161) = 186; C (167) = 186; C (185) = 184; C (186) = 182; C (188) = 181. Haciendo uso de los restantes parámetros (por ejemplo, mejora el uso de la CPU en un 97%), efectivamente se comprueba que la alternativa 10 es la óptima.

- Criterio 2 (*c2_p6b_toad*): coste de ejecución. La consulta original tiene un coste de 181, y el optimizador genera cuatro alternativas que lo mejoran. Empleando como único criterio el coste de ejecución, la alternativa 190 sería la más óptima, con un coste de 167 (un 8% mejor). Sin embargo, esto implica un empeoramiento de las condiciones respecto a la original, ya que tarda más tiempo (0,27 segundos en vez de 0,18) y emplea más recursos de CPU (305 vs 29). Esto significaría incurrir en un 50% más de tiempo y un 952% más de uso de CPU por mejorar un 8% los costes del plan. La relación negativa entre tiempo de ejecución y coste del plan es consistente entre las restantes alternativas.

c2_p5_toad, *c2_p7_toad* y *c2_p8_toad* continúan el análisis comenzado por *c2_p3_toad*, en el que se ponía a prueba la influencia de los niveles de inteligencia en el optimizador y el generador de índices sobre el rendimiento y mejora de las consultas.

Prueba *c2_p5_toad*: 15 horas, 712 escenarios, válidos 366. Se modifican los niveles de inteligencia del optimizador y el generador de índices a 5 y 2 respectivamente. No ha sido posible completar la exploración de posibilidades debido a restricciones temporales.

- Criterio 1 (*c2_p5a_toad*): tiempo transcurrido. Se logra una mejora temporal del 68’18% bajando de 0’22 segundos a 0’07 segundos en la alternativa 6. Esta incluye además un uso menor de CPU (de 0’40 a 0’14) y del plan, que pasa de 95 a 56.

- Criterio 2 (*c2_p5b_toad*): coste de ejecución. La alternativa 16 se posiciona como mejor alternativa de entre las disponibles, con un coste de 54 (en comparación a los 95 originales) y un tiempo de ejecución igual a 0,14 segundos (suponiendo una mejora en este aspecto del 36’36%). Esta es también la alternativa que menor tiempo de parse requiere: 0,03 segundos frente a los 0,19 segundos originales (es una mejora del 84’21% en este parámetro).

Prueba *c2_p7_toad*: 2h, 192 escenarios, válidos 156. En este caso los niveles de inteligencia cambian a: optimizador = 2 e índices = 5.

- Criterio 1 (*c2_p7a_toad*): tiempo transcurrido. En comparación a los 0’27 segundos originales, se han obtenido tres alternativas que lo disminuyen hasta 0’06 segundos. Aplicando como criterio secundario el coste, se tendrían las siguientes posibilidades: C (182) = 182; C (183) = 468 y C (188) = 181. El escenario 188 es además el que menos consumo de CPU tiene, 0’11. Por lo tanto, la mejor alternativa de las halladas por TOAD es la 188.

- Criterio 2 (*c2_p7b_toad*): coste de ejecución. El coste de la consulta original es 181, y la alternativa 8 lo reduce a 54. Esta también se ejecuta en un tiempo menor, 0’14 segundos, mejorando los resultados originales un 48’15%, y reduce a la mitad el consumo de CPU (de 0’5 a 0’25).

Prueba *c2_p8_toad*: Los niveles de inteligencia se sitúan en su máximo, 5. Se interrumpe el proceso de exploración de escenarios tras 19h en las que se generan 958 escenarios y se admiten como potenciales alternativas 587. Sin embargo, no todas ellas deberían considerarse, ya que en algunos casos los costes en términos de tiempo, coste del plan o uso de CPU son insostenibles. En concreto, la exploración en esta prueba ha incluido alternativas con un coste del plan del orden de millones:

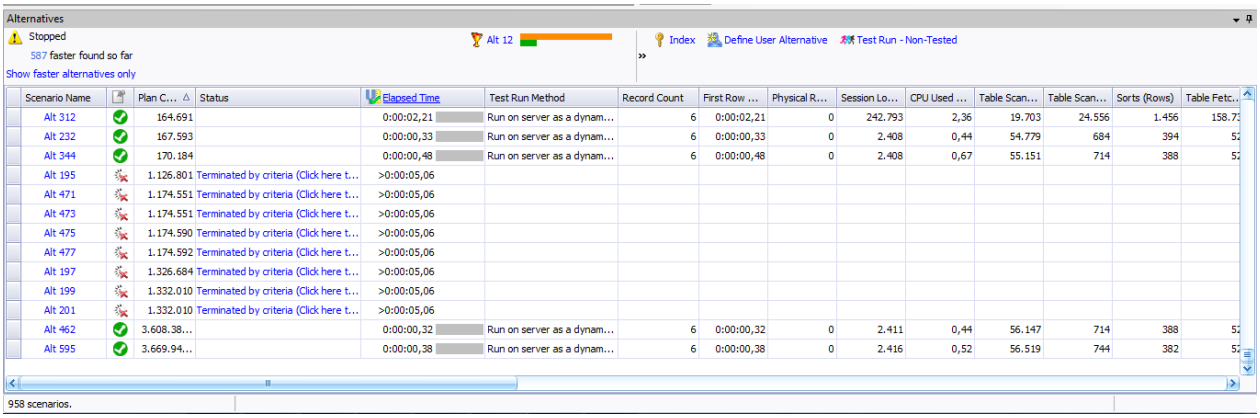


Ilustración 35: TOAD - Exploración escenarios

Centrando la búsqueda de alternativas en aquellas que minimizan ciertos parámetros, se han obtenido los siguientes resultados:

- Criterio 1 (*c2_p8a_toad*): tiempo transcurrido. Según este criterio, se obtiene una mejora del 81'25%, ya que se obtienen varias alternativas que reducen el tiempo de ejecución de 0'32 segundos a 0'06 segundos. En este caso surgen más de un escenario con este mismo tiempo, por lo que aplicando un segundo criterio, el coste de ejecución, se tiene que: C (12) = 182; C (32) = 58; C (34) = 181; C (37) = 219; C (179) = 182. Por lo tanto, la versión más optimizada según los criterios de tiempo y coste es la 32 (el original era 181), en la que también se reduce el consumo de CPU a 0'11 (frente al 0'72 original).

- Criterio 2 (*c2_p8b_toad*): coste de ejecución. Con un coste de 54, se han hallado dos posibles alternativas, la 16 y la 35. Teniendo en cuenta como criterios adicionales el tiempo de ejecución y el uso de CPU, la alternativa 35 es mejor: Tiempo (16) = 0'17 segundos vs Tiempo (35) = 0'15 segundos y CPU (16) = 0'33 vs CPU (35) = 0'29. El plan de ejecución de la alternativa 35 está formado por 79 pasos, mientras que el original sólo por 65, por lo que de nuevo se demuestra que la longitud del plan de ejecución no es un criterio válido para mejorar una consulta.

Se resumen las pruebas aplicadas así como los principales resultados en las siguientes tablas:

Consulta	ID_mejora	Descripción	Detalles
2	0a_toad	Ejecución inicial	Ejecución de partida, consulta sin modificar. Se esperan lecturas físicas.
2	0b_toad	Reejecución (se espera mejor resultado)	Segunda ejecución de la consulta original, será el punto de referencia. Objetivo: eliminar la
2	1_toad	Refactor > Convert to Oracle Join syntax	Sustituye los 'join - on' por 'from - where'
2	2_toad	Refactor > Convert to ANSI Join syntax	Sustituye los 'join - on' por 'from - inner join'
2	3a_toad	Optimize > Auto Optimize SQL + Advances Query Opt.	OptimIntelligence=2; IndexIntelligence = 2 (valores por defecto).
2	3b_toad	Optimize > Auto Optimize SQL + Advances Query Opt.	OptimIntelligence=2; IndexIntelligence = 2 (valores por defecto).
2	4_toad	Optimize > OEM	No es posible. Funcionalidad exclusiva bajo licencia Oracle adecuada.
2	5a_toad	Optimize > Auto Optimize SQL + Advances Query Opt.; Session Settings	OptimIntelligence=5; IndexIntelligence = 2. Alternativa con mejor tiempo.
2	5b_toad	Optimize > Auto Optimize SQL + Advances Query Opt.; Session Settings	OptimIntelligence=5; IndexIntelligence = 2. Alternativa con menor coste.
2	6a_toad	Optimize > Auto Optimize SQL	Criterio: tiempo.
2	6b_toad	Optimize > Auto Optimize SQL	Criterio: coste de ejecución.
2	7a_toad	Optimize > Auto Optimize SQL + Advances Query Opt.; Session Settings	OptimIntelligence=2; IndexIntelligence = 5. Criterio: tiempo.
2	7b_toad	Optimize > Auto Optimize SQL + Advances Query Opt.; Session Settings	OptimIntelligence=2; IndexIntelligence = 5. Criterio: coste de ejecución.
2	8a_toad	Optimize > Auto Optimize SQL + Advances Query Opt.; Session Settings	Optimizer Intelligence = 5; Index Intelligence = 5 (ambos al máximo). Criterio: tiempo.
2	8b_toad	Optimize > Auto Optimize SQL + Advances Query Opt.; Session Settings	Optimizer Intelligence = 5; Index Intelligence = 5 (ambos al máximo). Criterio: coste de ejecución.

Tabla 21: TOAD - Descripción pruebas Consulta 2

Consulta	ID_mejora	Coste	E-Rows	A-Rows	E-Time	A-Time	LRs	PRs	Optimizador
2	0a_toad	201	246	6	3 seg	1,345019 seg	1414	138	All Rows
2	0b_toad	181	246	6	3 seg	0,316224 seg	446	0	All Rows
2	1_toad	7950G	82T	279K	NS/NC	1,44 seg	716	0	All Rows
2	2_toad	2925G	30T	279K	NS/NC	1,2 seg	628	0	All Rows
2	3a_toad	182	246	6	3 seg	0,06 seg	712	0	All Rows
2	3b_toad	56	1	6	1 seg	0,15 seg	355	0	All Rows
2	4_toad								
2	5a_toad	56	1	6	NS/NC	0,07 seg	362	0	All Rows
2	5b_toad	54	1	6	NS/NC	0,14 seg	358	0	All Rows
2	6a_toad	181	1	6	NS/NC	0,06 seg	551	0	All Rows
2	6b_toad	167	1	6	NS/NC	0,27 seg	702	0	All Rows
2	7a_toad	181	246	6	NS/NC	0,06 seg	708	0	All Rows
2	7b_toad	54	1	6	NS/NC	0,14 seg	358	0	All Rows
2	8a_toad	58	1	6	NS/NC	0,06 seg	361	0	All Rows
2	8b_toad	54	1	6	NS/NC	0,15 seg	358	0	All Rows

Tabla 22: TOAD - Resultados Consulta 2

A la luz del conjunto de resultados obtenidos en las pruebas de la consulta 2, se han hecho los siguientes hallazgos:

- El hint '/*+ FULL(DEPARTMENTS) */' se ha empleado en pruebas que tienen por objetivo encontrar la alternativa más rápida, y fuerza al optimizador a realizr un scan completo de la tabla 'departments'. Aparece en las alternativas seleccionadas de las pruebas c2_p3a_toad y c2_p5a_toad, que sólo se diferencian en el orden en que se referencian las tablas.

- El hint '/*+ INDEX(REGIONS) */' aparece en la mejora c2_p5b_toad y permite ahorrar en recursos al acceder a los datos mediante un índice, e '/*+ INDEX_JOIN(REGIONS) */' en c2_p7a_toad y c2_p7b_toad, que sólo se diferencian en el orden de las tablas referenciadas en la cláusula from. El 'index_join' obliga al optimizador a ejecutar la operación join en base a un índice. Esto es posible sólo cuando un índice de pequeño tamaño contiene las columnas de datos necesarias. En ambos casos, los hints de índices se han aplicado sobre la tabla 'regions' porque al tener sólo cuatro valores y aparecer

con una frecuencia equilibrada, el acceso a los datos es más rápido. Si la distribución de valores fuera extrema, este no sería un hint válido porque para el valor más frecuente se acabarían realizando tantos accesos que saldría más costos que un full scan.

- Las pruebas c2_p8_toad también han incluido en las mejoras seleccionadas hints sobre los índices: '/*+ INDEX_DESC(LOCATIONS) */' en c2_p8a_toad e '/*+ INDEX_DESC(REGIONS) */' en c2_p8b_toad. El propósito de este hint es recorrer los índices en forma descendente, y esto es especialmente útil en casos en los que se calcula el máximo valor de una columna mediante la función MAX (que es lo que sucede en esta consulta).

Consulta 3:

Las cuatro primeras ejecuciones constituyen las pruebas *c3_p0a_toad*, *c3_p0b_toad*, *c3_p1_toad* y *c3_p2_toad*. Las dos primeras permiten establecer un punto de partida, de manera que son estos valores los que se pretenda reducir con el fin de optimizar la consulta. El optimizador de Toad por defecto, cada vez que realiza una operación de optimización, incluye también la ejecución de la sentencia original, lo que facilita la comparación al presentar tanto los datos originales como los alternativos en una sola pantalla. Las dos últimas comprueban si para la presente consulta refactorizar siguiendo los estándares de Oracle y ANSI ayudan a mejorar el rendimiento, como sí ha sucedido anteriormente.

En el caso de la consulta 3, los resultados de las cuatro pruebas iniciales son similares, debido a que sólo una pequeña porción de los datos se analiza, lo que evita lecturas en disco y cambios relevantes en las operaciones. En todos ellos, el coste del plan es de 50, los tiempos de ejecución son 0'09 o 0'1 segundos, y no hay lecturas físicas.

La ejecución de la prueba *c3_p3_toad* ha requerido 18 minutos para generar 99 escenarios y aprobar 59 como potencialmente elegibles. Niveles de los índices: 2 (Opt) - 2 (Índices).

- Criterio 1 (*c3_p3a_toad*): tiempo transcurrido. Cinco alternativas han sido ejecutadas en el menor tiempo posible, 0,03 segundos, que mejora un 70% el tiempo inicial (0,10 segundos). Adoptando el coste del plan como criterio secundario, se tiene para cada una de las cinco posibilidades los siguientes costes: C (3) = 62; C (4) = 45; C (5) = 50; C (7) = 50 y C (8) = 50. Por lo tanto, bajo estos criterios, la alternativa 4 supondría la más adecuada, ya que minimiza el tiempo y reduce ligeramente el coste de ejecución (la original tenía un coste de 50). Esta es también la alternativa de todas las generadas con menor uso de CPU: 0,04 (la original consume 0,24).

- Criterio 2 (*c3_p3b_toad*): coste de ejecución. Empleando el coste como primer criterio, además de la alternativa 4 comentada en el punto anterior se han generado otras cuatro posibilidades. Aplicando el tiempo como factor diferenciador entre ellas, se observan los siguientes resultados: t (4) = 0,03 seg; t (54) = 0,04 seg; t (91) = 0,11 seg; t (93) = 0,10 seg; t (95) = 0,11 seg. Así pues, para las pruebas *c3_p3_toad*, la misma alternativa es la óptima desde el punto de vista temporal como de costes.

Las pruebas *c3_p6_toad* optimizan la consulta 3 mediante la función Auto Optimize SQL. En 30 minutos ha extraído 98 alternativas posibles.

- Criterio 1 (*c3_p6a_toad*): tiempo transcurrido. Con una mejora del 73%, la alternativa 92 consigue bajar el tiempo de ejecución de 0,11 segundos a 0,03 segundos. No se requieren lecturas físicas, pero sí lógicas, 252. Aunque tanto la alternativa como la ejecución original tienen el mismo coste, 50, el escenario 92 conlleva un elevado uso de CPU: 65 frente a 2 (original). Por lo tanto, la aceptación de esta alternativa depende del entorno real en que se fuera a ejecutar la consulta.

- Criterio 2 (c3_p6b_toad): coste de ejecución. Atendiendo a este criterio, existen un gran número de alternativas que igualan (ninguna disminuye) el coste de ejecución original, 50. Incluyendo como criterio secundario el tiempo y en caso de empate el uso de CPU, se tendrían las siguientes alternativas: t (4) = 0,06 seg; t (5) = 0,04 seg; t (7) = 0,06 seg; t (8) = 0,06 seg; t (10) = 0,10 seg; t (24) = 0,04 seg; t (53) = 0,08 seg; t (54) = 0,08 seg; t (90) = 0,06 seg; t (92) = 0,03 seg; y t (95) = 0,10 seg. Por lo tanto, según el criterio de costes, la alternativa 92 vuelve a ser la óptima. En cambio, si se tuvieran en cuenta más parámetros, como el uso de CPU, se obtendría que la alternativa que mejor equilibra coste, tiempo y CPU de las disponibles es la 24, cuyo uso de CPU es sólo de 8, frente al 65 de la alternativa 92 o el 85 de la alternativa 5.

Prueba *c3_p5_toad*: 189 escenarios generados en 1’75 horas, de los cuales se consideran aptos 104, a pesar de haber interrumpido el proceso. Niveles de los índices: 5 (Opt) - 2 (Índices).

- Criterio 1 (*c3_p5a_toad*): tiempo transcurrido. Los 0,11 segundos originales se ven reducidos a 0,03 segundos en dos alternativas, cuyos costes son C (53) = 45 y C (188) = 50. Esto sitúa a la alternativa 53 como la óptima dados los presentes criterios.

- Criterio 2 (*c3_p5b_toad*): coste de ejecución. Se han hallado numerosas alternativas que reducen el coste de 50 a 45, entre las cuales se encuentra la alternativa 53 anterior. Utilizando el tiempo de ejecución como criterio extra, se tiene que: t (53) = 0,03 seg; t (4) = 0,10 seg; t (19) = 0,10 seg; t (23) = 0,11 seg; t (26) = 0,10 seg; t (34) = 0,14 seg; t (49) = 0,10 seg; t (120) = 0,10 seg; t (124) = 0,09 seg; t (125) = 0,09 seg y t (144) = 0,07 seg. Por lo tanto, la alternativa 53 se

convierte de nuevo en la óptima. Esta alternativa reduce además a un tercio el uso de CPU (de 0,24 originalmente a 0,08).

Prueba *c3_p7_toad*: En 18 minutos se exploran 99 escenarios, y 38 se consideran válidos por ser más veloces que la sentencia original. Niveles de los índices: 2 (Opt) - 5 (Índices).

- Criterio 1 (*c3_p7a_toad*): tiempo transcurrido. Aunque existen cuatro escenarios en los cuales el tiempo desciende hasta 0,03 segundos, uno de ellos, el 9, queda descartado automáticamente debido a su elevado coste (29.507). Utilizando el coste y el consumo de CPU como criterios secundario y terciario (porque también tienen el mismo coste, 50), se tendrían las siguientes alternativas: CPU (7) = 0,08, CPU (8) = 0,06 y CPU (10) = 0,10. Por lo tanto, la alternativa óptima bajo estos criterios es la 8. Esta gran similitud entre las alternativas 7 y 8 es debida a la posición del hint 'ordered' dentro de la sentencia SQL. En las consultas anidadas, la alternativa 7 lo coloca en el bucle externo, y por lo tanto el último en ser ejecutado según el plan; mientras que la 8 lo coloca en el bucle interno (concretamente en la subconsulta 1) y por lo tanto realiza la operación de ordenación antes y sobre un conjunto de datos inferior, de ahí que el consumo de CPU sea levemente mejor.

- Criterio 2 (*c3_p7b_toad*): coste de ejecución. Atendiendo a este criterio, hay tres alternativas con un coste igual a 45, que sin embargo se ejecutan en tiempos distintos y conllevan consumos de CPU diferentes. La alternativa escogida utilizando cualquiera de estos dos criterios es la 95: t (24) = 0,12 seg, t (53) = 0,09 seg, t (95) = 0,08 seg y CPU (24) = 0,23, CPU (53) = 0,19, CPU (95) = 0,15.

Prueba *c3_p8_toad*: En 9'5 horas TOAD genera 660 escenarios, de los cuales 387 mejorarían el tiempo de ejecución de la consulta original. Niveles de los índices: 5 (Opt) - 5 (Índices).

- Criterio 1 (*c3_p8a_toad*): tiempo transcurrido. Aplicando este criterio, la alternativa 98 minimizaría al máximo el tiempo de ejecución, pasando de 0,11 segundos originales a 0,02 segundos. Esta alternativa también es la que menos CPU consume, 0,05 en vez de 0,22.

- Criterio 2 (*c3_p8b_toad*): coste de ejecución. El coste de la consulta original es 50, y TOAD ha hallado cuatro alternativas que lo reducen a 42. Aplicando el tiempo y el uso de CPU como criterios secundario y terciario, se observan las siguientes alternativas: t (189) = 0,25 seg y CPU (189) = 0,34; t (290) = 0,21 seg y CPU (290) = 0,31; t (372) = 0,28 seg y CPU (372) = 0,36; t (475) = 0,19 seg y CPU (475) = 0,26. Ambos criterios señalan a la alternativa 475 como la más óptima entre las que menos coste de ejecución suponen. Las cuatro alternativas tienen en común haber empleado el hint 'first_rows(30)', y también las siguientes alternativas con menor coste. Esto tiene sentido, ya que al forzar el

cambio del optimizador a ‘first_rows’ se le está obligando a adoptar una meta de mejor tiempo de respuesta basado en un consumo mínimo de recursos.

A continuación se resumen las pruebas aplicadas así como los principales resultados en las siguientes tablas:

Consulta	ID_mejora	Descripción	Detalles
3	0a_toad	Ejecución inicial	Ejecución de partida, consulta sin modificar. Se esperan lecturas físicas.
3	0b_toad	Reejecución (se espera mejor resultado)	Segunda ejecución de la consulta original, será el punto de referencia. Objetivo: eliminar la
3	1_toad	Refactor > Convert to Oracle Join syntax	Sustituye los 'join - on' por 'from - where'
3	2_toad	Refactor > Convert to ANSI Join syntax	Sustituye los 'join - on' por 'from - inner join'
3	3a_toad	Optimize > Auto Optimize SQL + Advances Query Opt.	OptimIntelligence=2; IndexIntelligence = 2 (valores por defecto).
3	3b_toad	Optimize > Auto Optimize SQL + Advances Query Opt.	OptimIntelligence=2; IndexIntelligence = 2 (valores por defecto).
3	4_toad	Optimize > OEM	No es posible. Funcionalidad exclusiva bajo licencia Oracle adecuada.
3	5a_toad	Optimize > Auto Optimize SQL + Advances Query Opt.; Session Settings	OptimIntelligence=5; IndexIntelligence = 2. Alternativa con mejor tiempo.
3	5b_toad	Optimize > Auto Optimize SQL + Advances Query Opt.; Session Settings	OptimIntelligence=5; IndexIntelligence = 2. Alternativa con menor coste.
3	6a_toad	Optimize > Auto Optimize SQL	Criterio: tiempo.
3	6b_toad	Optimize > Auto Optimize SQL	Criterio: coste de ejecución.
3	7a_toad	Optimize > Auto Optimize SQL + Advances Query Opt.; Session Settings	OptimIntelligence=2; IndexIntelligence = 5. Criterio: tiempo.
3	7b_toad	Optimize > Auto Optimize SQL + Advances Query Opt.; Session Settings	OptimIntelligence=2; IndexIntelligence = 5. Criterio: coste de ejecución.
3	8a_toad	Optimize > Auto Optimize SQL + Advances Query Opt.; Session Settings	Optimizer Intelligence = 5; Index Intelligence = 5 (ambos al máximo). Criterio: tiempo.
3	8b_toad	Optimize > Auto Optimize SQL + Advances Query Opt.; Session Settings	Optimizer Intelligence = 5; Index Intelligence = 5 (ambos al máximo). Criterio: coste de ejecución.

Tabla 23: TOAD - Descripción pruebas Consulta 3

Consulta	ID_mejora	Coste	E-Rows	A-Rows	E-Time	A-Time	LRs	PRs	Optimizador
3	0a_toad	50	1	10	0,01 seg	0,09 seg	243	0	All Rows
3	0b_toad	50	1	10	0,01 seg	0,10 seg	243	0	All Rows
3	1_toad	50	1	10	NS/NC	0,10 seg	243	0	All Rows
3	2_toad	50	1	10	NS/NC	0,10 seg	243	0	All Rows
3	3a_toad	45	1	10	NS/NC	0,03 seg	243	0	All Rows
3	3b_toad								
3	4_toad								
3	5a_toad	45	1	10	NS/NC	0,03 seg	243	0	All Rows
3	5b_toad								
3	6a_toad	50	1	10	NS/NC	0,03 seg	252	0	All Rows
3	6b_toad	50	1	10	NS/NC	0,04 seg	246	0	All Rows
3	7a_toad	50	1	10	NS/NC	0,03 seg	350	0	All Rows
3	7b_toad	45	1	10	NS/NC	0,08 seg	243	0	All Rows
3	8a_toad	55	1	10	NS/NC	0,02 seg	272	0	All Rows
3	8b_toad	42	1	10	NS/NC	0,19 seg	1.868	0	All Rows

Tabla 24: TOAD - Resultados Consulta 3

Para esta consulta en concreto y tras analizar el conjunto de alternativas extraídas tras la batería de pruebas, se han encontrando los siguientes datos:

- Se vuelve a forzar el full scan, en este caso sobre la tabla ‘departments’ mediante el hint ‘/*+ FULL(DEPARTMENTS) */’ en las mejoras c3_p3a_toad y c3_p3b_toad (la alternativa según los dos criterios era la misma).

- En las pruebas c3_p5_toad, se emplea el hint ‘/*+ PX_JOIN_FILTER(Q1) */’, que utiliza un filtro Bloom para determinar si un elemento no pertenece a un determinado grupo. Los filtros de Bloom son útiles para determinar la no pertenencia de un elementos un grupo ya que se trata de una estructura poco consumidora de memoria. Sin embargo, hay que tener en cuenta que este filtro sólo es adecuado para detectar la no pertenencia, debido a que puede indicar

incorrectamente que un elemento sí está en un conjunto (es decir, no hay falsos negativos, pero sí falsos positivos). Esta propiedad es útil en casos como la consulta 3 sobre la subconsulta q1, donde la mayor parte de los empleados no habrá sido contratado en los últimos 4 años y se quiere descartar todos los que no cumplan esa restricción. Esta operación permite con una porción pequeña de memoria filtrar grandes cantidades de datos en un conjunto.

- Las pruebas c3_p8 muestran explícitamente el cambio de criterios entre las alternativas c3_p8a_toad y c3_p8b_toad. En la primera, el hint '/*+ USE_HASH(Q3,Q2) ORDERED */' busca obtener un salida de throughput o flujo adecuado, teniendo en cuenta los costes y el tiempo. Sin embargo, en la segunda, el hint '/*+ FIRST_ROWS(30) */' fuerza un cambio en la meta del optimizador, de forma que de prioridad a extraer las primeras 30 líneas de resultado minimizando al máximo los recursos. Y de ahí que las mejores alternativas consideradas en la prueba c3_p8b_toad tuvieran en común este hint.

5.4 Comparativa entre SQL Developer y TOAD

En esta subsección se han resumido las principales diferencias encontradas entre ambas herramientas durante el periodo de tiempo que se ha trabajado con ellas para evaluar el rendimiento de las tres consultas.

Aquellas diferencias que requieren una explicación más detallada que en la tabla son incluidas a continuación de esta y vienen marcadas con el signo '**'.

**	*SQL Developer*	*Toad*
1	Disponible en castellano	Sólo en inglés o chino
2	Menú de opciones en las tablas: columnas, datos, modelo, restricciones, permisos, estadísticas, disparadores, flashback, dependencias, detalles (metadatos), particiones, índices, sql (traducción sql de la definición y los datos de la tabla)	Menú de opciones en las tablas: columnas, índices, restricciones, disparadores, datos, script ('sql' en SQLDev), grants (privilegios, permisos), sinónimos, particiones, subparticiones, estadísticas y tamaño ('detalles' en SQLDev), referencias, 'usado por', políticas y auditoría.
3	Información de las columnas: nombre, tipo de dato, si puede o no ser nulo, valor por defecto, id de la columna, y comentarios.	Información de las columnas: nombre, id, si es PK (Primary Key), número de índices relacionados, si puede o no ser nulo, tipo de dato, valor por defecto, tipo de histograma, cuántos valores tiene distintos (NumDistinct), número de nulos, densidad, algoritmo de cifrado, 'Salt', 'Seq/Trigger', 'Virtual'.
4	Interfaz estática	Interfaz personalizable, puede ser reutilizada en el futuro si se guarda. Permite elegir la disposición de los

		elementos, la forma de presentación e incluso cuáles son visibles.
5**	Ver los resultados de distintas fuentes requiere usar más de una ventana y/o aplicación.	Desde la misma aplicación es posible observar las estadísticas de monitorización, el archivo trace asociado y los resultados de la ejecución.
6**	Presentación de los resultados amigable.	Presentación de los resultados en grandes cantidades, lectura difícil.
7**	Generación de tracefile bajo control.	Tracefile depende de Toad, no el usuario.
8**	Comparación bajo demanda y limitada a dos.	Comparación disponible incluso sin solicitarla, comparación múltiple.
9**	Ayuda limitada, y centrada en la guía oficial de la aplicación.	Ayuda disponible de distintas fuentes.
10	Coherencia de formato de los resultados.	Poca coherencia entre formatos según la función empleada para optimizar.
11	Visibilidad del tiempo empleado en ejecutar una acción, como generar un plan de ejecución, ejecutar una consulta, etc.	No hay feedback temporal, por lo que no es posible saber cuánto tiempo se ha empleado exactamente en realizar una acción.
12	No hay funcionalidad disponible gratuitamente para la optimización y mejora de consultas.	Prueba gratuita temporal para usar el optimizador de consultas.
13	Integración con Oracle perfecta: no requiere más permisos por ser ejecutado usando la herramienta.	Requiere permisos y configuraciones adicionales para integrarse con la base de datos Oracle.
14	Interpretación de los planes dependiente de los conocimientos del usuario.	Opción 'plain' para mostrar el plan, que hace más comprensible su interpretación.
15	Datos de registro básicos. Con esta misma cuenta se obtiene acceso tanto al software como a los manuales y white papers publicados.	Datos de registro detallados, necesarios tanto para descargar el software como para crear una cuenta en Dell y poder acceder a documentos de ayuda.

Tabla 25: Comparación SQL Developer vs TOAD

5** Como se verá posteriormente, la extracción de valores de parámetros empleados para valorar el rendimiento de cada ejecución tiene lugar utilizando varias fuentes. Esto conlleva abrir diversas aplicaciones y por lo tanto ventanas, lo que dificulta la labor de análisis del usuario. Toad en este aspecto es más eficiente, y mediante el uso de pestañas permite al usuario navegar por los distintos tipos de datos extraíbles de la ejecución:

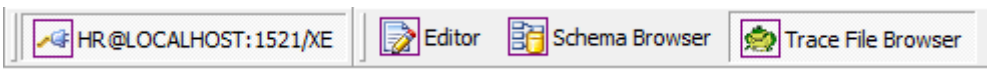


Ilustración 36: TOAD - Pestañas de navegación

6** Desde el punto de vista de la usabilidad, la presentación de los resultados en ambas herramientas difiere en gran medida por la presencia o ausencia de colores. En

SQL Developer los planes de ejecución eran presentados sobre un fondo rayado que permitía diferenciar rápidamente los resultados de diferentes filas. Además, los datos de las estadísticas eran acumulados verticalmente, mientras que en Toad, al incluir datos de varias sentencias al mismo tiempo, se muestran de forma horizontal (y en términos de usabilidad es peor elección).

7** En SQL Developer, mediante el uso del script era posible determinar cuándo se generaba el archivo tracefile y asegurar que su contenido contenía exclusivamente datos relacionados con la última ejecución de la consulta correspondiente. En el caso de Toad, este control no está en manos del usuario, por lo que sólo es posible utilizar el navegador de tracefile, y que incluye datos de otras sentencias SQL. En este caso, para seleccionar los datos de una sentencia en concreto basta con pulsar sobre ella, pero los archivos no podrán estar diferenciados como sucedía en SQL Developer.

8** La comparación del rendimiento entre ejecuciones se realizaba mediante la indicación explícita en SQL Developer. Esto suponía generar los archivos de los cuales se extraían los parámetros de rendimiento, marcarlos (‘pin’) y seleccionar dos ejecuciones para comparar. Toad, en su navegador tracefile, muestra una gran tabla en la que cada fila está dedicada a la ejecución de una sentencia, por lo que en caso de requerir comparar resultados basta con acceder a esa pestaña, y además el número de elementos a comparar no está restringido a dos.

9** En caso de duda acerca del significado de un elemento o un concepto, SQL Developer cuenta con un acceso integrado a su documentación oficial. Sin embargo, la información incluida en este manual no es detallada y en ocasiones queda falto de explicaciones, lo que obliga a consultar fuentes externas no oficiales. Toad por otro lado, en cada error surgido muestra la opción ‘jump’ que conduce a todas las categorías de ayudas disponibles: documentación oficial en Toad World y Dell, foros oficiales de toad, release notes y escribir una pregunta.

6. Discusión final [*Final discussions*]

Esta última sección está dirigida a valorar el estudio en su completitud, haciendo especial énfasis en los principales hallazgos y conclusiones, las limitaciones bajo las cuales se ha desarrollado y que han influido en los resultados, para luego convertirse en futuras vías de ampliación de este trabajo.

Para ello se han delimitado tres apartados claramente diferenciados, escritos tanto en español como en *inglés*, con el fin de que sea accesible para interesados que no necesariamente dominen el castellano.

Final Discussions

The aim of this final section it to offer a deeper evaluation of the project now ended. Its contents will be divided into three parts that will be focused on three distinct areas: in first place, the main conclusions and findings reported in the document will be summarized; afterwards the limitations and restrictions that have affected the project will be enumerated; and finally, a list of potential future works that could continue this investigation has been added.

All these paragraphs have been written both in Spanish and English so this contents are available not only to Spanish speakers, but to everyone who may be intestested on it.

6.1 Conclusiones [*Conclusions*]

La elección de optimización de consultas como materia de estudio ha brindado la oportunidad de llevar a cabo en un solo proyecto una gran variedad de investigaciones que han permitido obtener una visión actualizada de los sistemas gestores de bases de datos relacionales, trabajar con dos de las herramientas gráficas de gestión más extendidas así como mejorar la comprensión y capacidad analítica del rendimiento de las consultas SQL.

Durante el último año se han consultado manuales, foros, blogs de opinión, webs oficiales de las empresas, informes publicados por especialistas y material didáctico de instituciones educativas, todo ello para progresivamente profundizar en el inicialmente escaso conocimiento relacionado con la optimización de consultas.

Se ha comprobado que no existe una solución absolutamente perfecta, o superior al resto, sino que las decisiones que se toman para escoger las herramientas de trabajo están influidas con un amplio abanico de factores. A la cabeza de esta lista se encuentran las restricciones presupuestarias, el tipo de entorno en que se emplee la base de datos y su enfoque a rapidez, disponibilidad o mantenimiento de la coherencia, el

nivel de conocimiento del usuario administrador, y la curva de aprendizaje que este esté dispuesto a asumir.

Si bien es cierto que a pesar de la amplia oferta de posibilidades el software propietario sigue contando con más adeptos que el libre, este último poco a poco está mejorando su posición. Este cambio no es exclusivamente mental, por el cual los individuos deciden invertir en un producto nacido de la colaboración voluntaria, sino que también el descontento por las elevadas tasas en las licencias ha influido en ello. En la sección dedicada a la oferta se añadió un apartado en el cual se consideraban incluir para el estudio el pago de las licencias oportunas. Al ser tan elevadas, funcionan como barreras de entrada para aquellas empresas que podrían beneficiarse de su uso pero no cuentan con los medios necesarios.

La formación también es un importante factor a la hora de trabajar en la optimización de las bases de datos. El sector de las tecnologías de la información crece y evoluciona a tal velocidad que sin el apoyo debido por parte de los impulsores de estas novedades resulta una tarea muy laboriosa y difícil de llevar a cabo el mantenerse constantemente actualizado sobre las novedades. Cuanto más se retrasa o se subestima esta necesidad, mayores son los costes para la organización: un sistema antiguo es un sistema vulnerable, no invertir en la formación de los trabajadores puede ser motivo de abandono, y el consumo innecesario de recursos contradice los principios básicos de una empresa, cuyo fin es crear riqueza, no desperdiciarla.

Las licencias de software no sólo afectan a las empresas, sino a interesados como el grupo objetivo de este estudio, los alumnos universitarios, que no pueden desarrollar completamente sus conocimientos técnicos cuando son ellos quienes supondrán el relevo de los actuales profesionales. TOAD limita la curva de aprendizaje debido a la finalización del periodo de prueba, aunque gracias a él se ofrece la oportunidad de interactuar mínimamente con la herramienta. En el caso de SQL Developer esta oportunidad no está presente en ningún momento, por lo que el aprendizaje está restringido únicamente a la gestión y configuración básica de una base de datos.

Aplicar ambas herramientas ha permitido descubrir aspectos beneficiosos de cada una de ellas, y que pueden ser explotados si se continúan utilizando como complementarias. Por un lado, de SQL Developer se ha aprendido a analizar el rendimiento de una sentencia SQL aunque para ello sea necesario consultar más de un archivo, ya que muestra siempre (si se configura adecuadamente) el plan de ejecución con toda la información, estimada y real. Esto no ha sucedido con TOAD, que ha requerido ejecutar más de una vez la consulta para extraer el plan de ejecución y a pesar de ello no muestra algunos de los parámetros más importantes (como la estimación del tiempo).

Por otro lado, las tareas de optimización y mejora de consultas recaen en TOAD tanto por dar acceso libre a ellas (aunque sea de forma temporal) como por el potencial que tiene la herramienta en cuanto a diversidad de funciones y calidad del producto. Su utilidad es tal que sigue siendo recomendable a pesar del reducido periodo de prueba, el

extenso consumo de recursos realizado en sus búsquedas de escenarios y las limitaciones encontradas sobre ella.

Tras las pruebas realizadas en SQL Developer y TOAD de los tres casos de uso presentados en la sección 5, se pueden destacar las siguientes conclusiones:

- El número de pasos del plan de ejecución no determina el coste que pueda tener. Como se ha podido observar en las sucesivas pruebas, el número de pasos en los que se ejecuta un plan no es indicador del rendimiento que tendrá una consulta. Por ejemplo, en la consulta 3, las pruebas c3_p3_toad y c3_p8a_toad tienen 24 pasos en cada uno de sus planes de ejecución, pero las primeras tienen un coste de 45 mientras que la última 55. El tiempo tampoco está fuertemente relacionado con el número de pasos, ya que se han encontrado pruebas con tiempos de ejecución muy distintos a pensar de haber seguido el mismo número de pasos.

- En la mayoría de los casos la mejora del coste de ejecución conlleva un gran aumento en el tiempo transcurrido y un uso más intensivo de la CPU, por lo que suele ser más recomendable mejorar los tiempos de ejecución (el % de mejora del tiempo es superior al % de desmejora en el coste).

- Las reducciones de tiempo requieren más operaciones de lectura lógicas.

- Los niveles de inteligencia influyen en la ejecución de la sentencia original.

- A mayor tamaño de muestreo, mejores estimaciones y resultados de ejecución, ya que el optimizador cuenta con más información a partir de la cual diseñar un plan de ejecución adecuado para la sentencia SQL que se va a ejecutar.

- Un aumento del valor de fetch rows hace que la cantidad de información trasladada en cada transacción sea superior, por lo que el número de llamadas decrece.

- Conocer el estado de las tablas de datos, el entorno en que se ejecutarán las consultas y otra información que el sistema no es capaz de saber pero sí el administrador sirve para elegir adecuadamente los hints. Aunque estos no sean recomendados por Oracle, en determinados casos en los que el usuario sepa que forzando al optimizador a actuar de una forma determinada actuará de manera más eficiente pueden ser muy beneficiosos. Un ejemplo de ello es la utilización del filtro Bloom visto en la consulta 3 [5.3 Optimización de consultas en TOAD].

Se han analizado SQL Developer y TOAD como dos herramientas rivales, competidoras entre sí, pero a la luz de los resultados y el análisis de las respectivas ventajas e inconvenientes encontrados a durante su fase de prueba, lo más óptimo sería aprovechar las ventajas de ambas herramientas y complementarlas entre sí.

Por un lado, SQL Developer aporta una plataforma sencilla de instalar y usar, integrada perfectamente con las bases de datos Oracle, y desde la cual la extracción de los resultados de las ejecuciones y visualización de los parámetros de rendimiento es menos confusa que TOAD. Disponer del control de creación de cada archivo tracefile por separado es una de las grandes ventajas de SQL Developer cuando se pretende analizar una única ejecución y aislarla de las demás.

Por otro, TOAD aporta durante 30 días (o más, si se asumen los costes de licencia) un motor de optimización potente que puede ser manejado por usuarios tanto expertos como inexpertos, gracias a una interfaz gráfica amigable. De las tres vías de optimización, las dos utilizadas (Auto Optimize SQL y Advance SQL Optimizer) han demostrado ser igualmente válidas en la generación de alternativas, aunque con diferencias: Auto Optimize SQL recaba las alternativas en menor tiempo, pero a cambio genera menos y no permite al usuario aplicar cambios detallados en la configuración de este proceso. En cambio, Advance SQL Optimizer sí es capaz de definir con más detalle qué aspectos se desea tener en cuenta durante la generación, qué hints incluir, formatos, número máximo de elementos generados, etc; todo ello para abarcar un número más amplio de alternativas, lo que conlleva más tiempo en el proceso (algunas ejecuciones han superado las 10 horas).

En conclusión, este documento ha cumplido los objetivos establecidos al comienzo, aportando una visión analítica y detallada de dos de las herramientas más comunes para la gestión de bases de datos, y que queda a disposición de todo aquel interesado en la materia.

Conclusions

Having selected queries optimization in relational database management systems has given the opportunity to conduct in a unique project a wide variety of investigations which have given an updated view of those systems, the opportunity to work with two of the most used tools by professional of this industry, and improve comprehension and analytical skills when evaluating queries performance.

Last year has been dedicated to retrieve and gather relevant information from multiple sources of information that include white papers published by companies, official knowledge bases, documents, release notes, online tutorials, official forums and blogs, with the purpose of reaching increasingly more detailed information about the topic. The more information was retrieved, the more I realized how extended was this theme.

From the comparisons performed between relational and no relational databases, between relational management systems and then between graphic tools one thing is sure: there is not a perfect solution that can solve every problem a company or a user

may face when trying to manage their database. There are a huge number of aproximations that might be adequated in certain environments but not for everyone.

Instead, when deciding which one of all the espectrum of possibilities should be chosen, there is a list of factors that affect it:

- *Budget restrictions.Section [*2.2 Planificación Inicial: GANTT y Oferta*] showed an alternative offer which included in the price the costs of paying certain licenses. Only some companies might own the required assest to pay for them, but even them have a maximum budget. Investing on these expensive systems is part of the financial strategy of the company, it is about resources allocation, so it is not easy to accept it.*

- *The environment in which a database will be deployed and used, which includes the kind of queries it may have to answer, the goals the optimizer should pursue, and the workflow and throughput it may have to manage. Some databases are required to be fast, others to be always available and others to maintain consistency.*

- *The knowledge level of the administrator or user is also one of the main factors. A novice user would rarely pay for an expensive system without knowing how to manage it.*

- *Also related to the administrator or user, the learning curve he is up to accept in order to learn something new will affect the decision because if there is no interest from his side, it doesn't matter how well documented is the new system, he won't take advange of it.*

Truth to be told, private software is still preferred compared to free software, even though these last years the quantity released of this kind of software has increased. Not only is there more free software possibilities, they are becoming more used in enterprises. However, this is not only because of a change in mentality, for which individuals work together and collaborate in order to achieve a common goal, this tendency has its roots in the last economic crisis, where hundreds of business had to reduce costs and they are still doing. So, like said before…budget restrictions. When costs related to some products are so high, they then function as an entrance barrier that prevents potential clients from using it.

Training is also an important factor when working on optimizing databases. The Information Technology industry grows and evolves so fast that without the support of the drivers of these improvements it is a very laborious and difficult task of carrying out the keep constantly updated. The more it is delayed or this need is underestimated, the greater the costs for the organization: an old system is a vulnerable system, not investing in internal training of their workers may lead to abandonment, and unnecessary consumption of resources contradicts the basic principles a company, whose purpose is to create wealth, not waste it.

Software licenses not only affect business but also to stakeholders as the group target of this study: college students, who can not fully develop its expertise when it is them who will involve over from the current practitioners. TOAD limits the learning curve due to the end of its trial period, but thanks to it at least offers the opportunity to interact minimally with the tool. In the case of SQL Developer this opportunity is not present at any time so learning is restricted to the usage of certain functionalities related to management and basic configuration of a database.

To have being able to use both tools has helped to uncover beneficial aspects of each of them, that can be taken advantage of if they continue to use as complementary instad of rivals. On the one hand, SQL Developer has permitted to learn to analyze the performance of a SQL statement even if it is necessary to consult more than one file, as it shows always (if properly configured) the execution plan with all the detailed information, both estimated and real. This has not happened with TOAD, which required to run more than once the same query when trying to retrieve the query execution plan and although it does not show some of the most important parameters (such as time estimations).

On the other hand, the tasks of optimization and improvement of queries are owned by TOAD, which gives free access to them (unfortunately only temporarily) and has an inmense potential thank to its diversity of functions and product quality. Its usefulness is such that it is still advisable despite the small trial, the extensive use of resources made their searches scenarios and limitations found on it.

After having perfomed several tests on TOAD and SQL Developer of the three use cases presented in 5.1 Diseño de los casos de uso*, we can highlight the following conclusions:*

- *The number of steps of the execution plan does not determine the cost an execution may have. As observed in successive tests, the number of steps in which a plan is performed it is not an indicator of a query performance. For example, in the query 3 the c3_p3_toad and c3_p8a_toad tests needed 24 steps in each of their implementation plans, but the firsts had a cost of 45 while the last 55. Time of execution is not strongly related to the number of steps because evidences were founded in which with very different execution timings were the result of the same number of steps.*

- *In most cases the cost of improvement implementation entails a large increase in time and use of the CPU, so it is often more advisable to improve execution time instead of other performance parameters (this is, the % time improvement is higher the % of deterioration in the cost).*

- *Reductions in timing require more number of logical reads.*

- *The intelligence levels influence the execution of the original query, as they help to explore a wider range of potential scenarios that might be better. However, only the 'optimizer intelligence level' has been proved to affect the optimization process, because it only involved query rewrites, no index optimization.*

- *The higher is dynamic_sampling_size, the better results are achieved. The optimizer depends on the information it may have about the objects involved in the SQL sentence which is going to be optimized, so when it gets more informationit can design a better adapted execution plan, which ends in better performance.*

- *When the value of fetch rows increases, the number of user calls decrease, because on each transaction more information is retrieved.*

- *Knowing the status of the data tables, the environment in which queries are executed and other information that the system is not capable of knowing but the administrator does helps to properly choose the hints. Although these mechanisms are not recommended by Oracle, in certain cases when the user knows that forcing the optimizer to act in a certain way it will act more efficiently can be very beneficial. An example of this is the use of the Bloom filter seen in consultation 3[*5.3 Optimización de consultas en TOAD*].*

We analyzed SQL Developer and TOAD as two rival tools, competing with each other, but in light of the results and analysis of the respective advantages and disadvantages found during its test phase, the most recommendable approach would be to take advantage of both tools and use them as a complement of each other.

On the one hand, SQL Developer provides a simple platform to install and use, integrated seamlessly with Oracle databases, and from which the extraction of the results of executions and display performance parameters is less confusing than TOAD. Having control over the creation of each tracefile file separately is one of the great advantages of SQL Developer when trying to analyze a single execution and isolating it from the others.

On the other, TOAD provides during 30 days (or more if licensing costs are assumed) a powerful optimization engine that can be handled by both experts and inexperienced users, thanks to a friendly graphical interface. Of the three ways of optimization, the two used (Auto Optimize SQL and Advance SQL Optimizer) have proven to be equally valid in generating alternatives, but with remarkable differences: Auto Optimize SQL collects alternatives in less time, but in return it generates lessscenarios and it does not allow the user to apply detailed changes in the configuration of this process. On the contrary, SQL Optimizer Advance itself is able to define in more detail what aspects must be considered during this alternatives generation, which hints should include, formats, maximum number of generated elements, etc; all to cover a wider number of alternatives, leading to more time spent in the process (some executions have exceeded 10 hours).

In conclusion, this document has met the objectives set at the beginning, providing an analytical and detailed view of two of the most common tools for managing databases, and is available to anyone interested in the subject.

6.2 Limitaciones [*Limitations*]

La aceptación de los resultados y las conclusiones anteriormente expuestos están condicionados por los recursos del sistema (véase CPU, I/O, memoria, arquitectura de la BD, etc), la distribución de los datos, la arquitectura del sistema, los planes de ejecución obtenidos y el uso que le de el usuario (en este caso, pruebas orientadas a la optimización de las consultas), por lo que las conclusiones extraídas no son aplicables universalmente. De estas limitaciones de las herramientas y los recursos disponibles, pueden establecerse potenciales vía de desarrollo futuras, como se verá en 6.3 Trabajos futuros [*Future works*].

A continuación se enumeran aquellas con mayor grado de influencia sobre el rendimiento de las ejecuciones y que deben ser tenidas en cuenta a la hora de aceptar las conclusiones:

- El RDBMS empleado ha sido la edición exprés de Oracle 11g versión 2. Aunque en lo relativo a requisitos de recursos e facilidad de instalación fue la mejor opción, la imposibilidad de ejecutar las operaciones en paralelo y no disponer de acceso a la caché de resultados han supuesto una limitación que podría evitarse empleando la versión estándar o enterprise.

- Restricciones temporales. La prueba gratuita proporcionada por Dell para su software Toad for Oracle es válida únicamente durante 30 días. La curva de aprendizaje básico puede completarse en dicho período, pero el potencial de esta herramienta requeriría más tiempo. En el ámbito de las optimizaciones, también han surgido restricciones temporales dado que en ocasiones la búsqueda de alternativas ha consumido un gran número de horas, con el consecuente gasto de recursos de CPU y electricidad para llevarla a cabo. Ello ha causado que se decidiera imponer un intervalo máximo de tiempo para explorar nuevos escenarios, aunque no se llegase a completar esta operación.

- El hardware empleado corresponde a un equipo portátil como se expuso al comienzo del estudio. La elección de este equipo ha sido tenía por objetivo simular en el entorno de pruebas la realidad, y es que las tareas de optimización no siempre se aplican sobre sistemas más preparados. Este estudio ha permitido demostrar que a pesar de esta restricción las optimizaciones son posibles.

- Restricciones presupuestarias y licencias. El análisis y la optimización en SQL Developer están basados en la modificación manual de parámetros estudiados en

los manuales oficiales proporcionados por Oracle u observados gráficamente en la configuración de SQL Developer ya que los paquetes dedicados a la optimización y mejora de consultas sólo están disponibles bajo licencia y al contrario que Toad no cuentan con prueba gratuita.

- Los refactorizados no siempre traducen correctamente. Por ejemplo, la versión original de la consulta 2 y la versión refactorizada c2_p1_toad no obtienen los mismos resultados, de ahí que los valores de los parámetros fueran tan diferentes respecto a las restantes pruebas.

Limitations

The acceptance of the results and conclusions above presented are conditioned by the system resources (see CPU, I / O, memory, architecture BD, etc.), data distribution, system architecture, implementation plans obtained and the use that gives the user (in this case, oriented query optimization tests), so these conclusions should not be considered as universally applicable. From these limitations of the tools and resources available, potential future works guidelines can be established, as it will be said in 6.3 Trabajos futuros [*Future works*].

Below are listed those restrictions with greater degree of influence on the performance of executions and that should be taken into account when accepting the conclusions:

- *The RDBMS used has been Oracle Express Edition 11g Release 2. Although in terms of resource requirements and ease of installation was the best option, inability to perform operations in parallel and have no access to cache results have impossed a limitation that could be avoided by using the standard or enterprise editions.*

- *Time restrictions. The free trial provided by Dell for Toad for Oracle software is only valid for 30 days. Basic learning curve can be completed in that period, but the potential of this tool would require more time. In the field of optimizations they have also appeared time restrictions because sometimes the search of alternatives has consumed a large number of hours, with the respective spending of CPU resources and power to carry it out. This has caused to decide to impose a maximum time to explore new scenarios, even if it did not reached the end of this operation.*

- *The used hardware corresponds to a laptop as discussed at the beginning of the study. The aim of this choice of this equipment was to simulate reality in the test environment, because truth is that optimization tasks are not always applied on fully prepared systems. This study has demonstrated that despite this restriction optimizations are possible.*

- *Budgetary and licensing restrictions. Analysis and optimization in SQL Developer are based on manual modification of parameters studied in the official manuals provided by Oracle or observed graphically in configuring SQL Developer since packages dedicated to the optimization and upgrading of consultations are only available under license and unlike Toad they do not provide free trial.*

- *The refactored variations were not always translated correctly. For example, the original version of the query 2 and the refactored version c2_p1_toad do not get the same results, hence the parameter values were so different respect to the remaining tests.*

6.3 Trabajos futuros [*Future works*]

A raíz de los resultados obtenidos y las limitaciones encontradas a lo largo del proyecto, se proponen a continuación potenciales líneas de ampliación del estudio:

- Emplear una nueva base de datos o ampliar la utilizada para adaptarla a un entorno más real. Esto significaría contar con más tablas integradas con las ya existentes, así como un volumen mayor de datos, todo ello para obtener un entorno de pruebas que sea capaz de simular con mayor realismo las condiciones a las que se enfrentaría un administrador en el mundo real. Un ejemplo de estas grandes bases de datos serían las manejadas por los software ERP (*Enterprise Resource Planner*), en las que se integran numerosos y diversos datos de la empresa con el fin de mejorar las decisiones gracias la disposición completa de información.

- Ampliar el número de casos de uso para explorar optimizaciones en consultas que hagan uso de más estructuras. Sería particularmente interesante emplear uniones, los diferentes subtipos de join y operaciones sobre los datos, ya que aunque la consulta 2 involucraba MAX, MIN, y AVG estas no constituyen una muestra representativa.

- Repetir el estudio incluyendo técnicas de paralelización. Las pruebas en serie realizadas en el estudio tenían por objetivo comprobar qué grado de optimización podría obtenerse dadas las limitaciones de hardware y procesamiento sólo serial. Sin embargo, una ampliación interesante sería comprobar cuáles serían los efectos resultantes de la combinación simulátena de las técnicas de optimización junto a la ejecución paralela. Entornos en los que la rapidez de ejecución es crítica, véase transacciones financieras de alta velocidad, se verían beneficiados.

- Repetir el estudio utilizando una edición Oracle que incluya un abanico más extenso de funcionalidades y características, ya que la versión exprés tiene

muchas de ellas restringidas. Un ejemplo sería evaluar el efecto que el control sobre la caché de resultados, incluido en la edición de empresa, tendría sobre la optimización de consultas.

- Empleando una base de datos que almacene un mayor número de datos, reejecutar la batería de pruebas elaborada para SQL Developer y así comprobar si la falta de diferencias entre resultados es debido al tamaño de la base de datos con la que se trabaja o a la independencia de la modificación de dichos parámetros sobre el rendimiento de la consulta.

Future Works

Following the results achieved and constraints encountered throughout the project, proposed below potential lines extending the study:

- *Use a new database or expand the one used to fit a more real environment. This would mean having more tables integrated with the existing as well as a greater volume of data, all of these in order to create a test environment that is able to simulate more realistic conditions to which an administrator would face in the real world. An example of these large databases would be the ones usually managed by ERP software (Enterprise Resource Planner), where numerous and diverse enterprise data are integrated in order to improve decisions with full information available.*

- *Expand the number of use cases to explore optimizations for queries that make use of more structures. It would be particularly interesting to use unions, different subtypes and join operations on the data, since although the request 2 involved MAX, MIN, and AVG these do not constitute a representative sample.*

- *Repeat the study including parallelization techniques. Serial testing conducted in the study was designed to test what degree of optimization could be achieved given the limitations of hardware and only serial processing. However, an interesting extension would be to see what would be the resulting effects of simultaneous combination of optimization techniques with parallel execution. Environments where execution speed is critical, see high-speed financial transactions, would benefit.*

- *Repeat the study using an Oracle edition including a wider range of capabilities and features, as Express version has many restricted. An example would be to assess the effect over the 'results_cache', including in the enterprise edition, which would have on query optimization.*

- *Using a database to store more data, rerun the test battery developed for SQL Developer and check whether the lack of differences between the results is due to the size of the database with which these testing was performed or is it because of the no effect of modifications over these parameters on the performance of the query.*

7. Glosario

Access Path (método de acceso): método según el cual se accede y se recuperan los datos de una base de datos. Los métodos de acceso más comunes son seis: full scan tables, rowid scan, index scan, cluster access, hash access, y sample table scan. (Oracle DB Performance Tuning Guide - Capítulo 11)

Aislamiento (ACID): los cambios provocados por una transacción deben ser independientes de otras transacciones.

Atomicidad (ACID): cada transacción es 'todo o nada', es decir, si parte de la transacción fallase, la transacción completa se consideraría fallida y por lo tanto no se cambiaría el estado de la base de datos.

Base de datos: es una forma de almacenamiento de datos que permite extraer información de ella. Se dividen esencialmente entre bases de datos relacionales y no relacionales.

Base de datos no relacional: Representan los datos en colecciones de documentos JSON (Java Script Object Notation), y sus datos no están estructurados.

Base de datos relacional: Es el enfoque más simple. En estas bases de datos la información es almacenada en tablas con filas y columnas, de manera que una tabla es concebida como un conjunto de objetos del mismo tipo (un conjunto de filas). Están regidas por las reglas de integridad y sus operaciones se basan en el álgebra relacional.

Combinación AP (CAP): los nodos se mantienen en línea incluso cuando no es posible la comunicación entre ellos. Una vez la partición se haya resuelto, estos se resincronizan, pero no se garantiza que todos los nodos contengan los mismos datos tanto antes como después de la partición. Este es el enfoque de las bases de datos no relacionales.

Combinación CA (CAP): los datos son consistentes entre todos los nodos. Además, mientras estén en línea, los datos introducidos en uno de ellos son replicados en el resto de nodos, por lo que los datos son los mismos en todos ellos. Esta es la combinación escogida por las bases de datos relacionales.

Combinación CP (CAP): los datos son consistentes entre todos los nodos y sólo queda en estado no disponible cuando uno de los nodos falla, lo que permite ser tolerante a particiones.

Consistencia (ACID): la base de datos parte de un estado consistente antes de comenzar una transacción y también llega a un estado consistente, haya sido o no un éxito la transacción.

Consistencia (CAP): una vez se ha escrito un dato, todas las futuras lecturas contendrán dicho dato.

Disponibilidad (CAP): la base de datos debe estar siempre disponible.

Durabilidad (ACID): una vez realizada la transacción y se notifica al usuario, no se podrá deshacer dicha transacción.

'*Fetch*' o '*Fetching*': es el proceso de devolución de líneas como resultado al cliente. En peraciones de insertado, actualización, y borrado, no hay nada que devolver, por lo que en estos casos el valor será 0 en el informe TKPROF. Incluso es posible obtener un 0 en sentencias select, pero esto significaría que no había resultados que devolver.

Heurísticas de Nielsen para la mejora de la usabilidad de las interfaces gráficas: 1 – Visibilidad del estado del sistema; 2 – Correspondencia entre el sistema y el mundo real; 3 – Libertad y control del usuario; 4 – Consistencia y estándares; 5 – Ayudar al usuario a reconocer, diagnosticar y recuperarse de errores; 6 – Prevención de errores; 7 – Reconocimiento antes que recuerdo; 8 – Flexibilidad y diseño minimalista; 9 – Estética y diseño minimalista; 10 – Ayuda y documentación.

IDE o Integrated Development Environment: conocido en español como 'entorno integrado de desarrollo', es un software que contiene las principales herramientas de que un desarrollador necesita para desarrollar y probar un software. Los componentes más comunes son un editor de código, un compilador o interpretador y un 'debugger', accesibles mediante una interfaz de usuario.

Lenguaje SQL: Lenguaje de consulta de base datos cuyas siglas se correspoden con "Structured Query Language", diseñado para ser usado en bases de datos relacionales. Aunque existen algunas variantes, hay ciertos comandos comunes al estándar que son empleados en todos los RDBMS, como la cláusula 'select'.

LIO o Logical Input/Output: Son lecturas y escrituras realizadas sobre el buffer caché. Para que esto suceda, antes debe haberse producido un PIO.

Oracle APEX: *Oracle Application Express*. Es una herramienta Oracle de desarrollo rápido que permite diseñar, desarrollar e implantar aplicaciones responsive sobre una base de datos utilizando el navegador web. Web oficial: https://apex.oracle.com/es/.

'*Parse*' o '*parsing*': es un concepto similar a 'compilación' en software, referido a la actividad de procesar un código para convertirlo en ejecutable. En este caso lo que convierte en ejecutable es la sentencia SQL. El proceso consta de dos fases: 1) análisis sintáctico y 2) análisis semántico. Existen dos tipos de parse: soft y hard parse.

PIO o Physical Input/Output: Son lecturas y escrituras físicas, lo que significa que se han realizado sobre el disco. Una vez se recupera información de disco, se guarda en la caché, y si se vuelve a solicitar la misma información sólo será necesario un LIO para recuperarla. Tiene un coste superior al LIO.

PGA o Program Global Area: Región no compartida de memoria que contiene información de datos y control. Existe un PGA por cada proceso, y el conjunto de todos

ellos se conoce como ‘instancia PGA’. Es el valor de este último (‘`pga_aggregate_target`’) el que puede establecerse, no el individual. Aparecerá en las estadísticas memstats cuando se trate de operaciones intensivas en memoria, como ‘sort’, ‘hash join’, ‘bitmap merge’ o ‘bitmap create’. El valor del parámetro ‘`pga_aggregate_target`’ afecta al parámetro ‘`workarea_size_policy`’.

Propiedades ACID: Atomicidad, Consistencia, Aislamiento y Durabilidad. Definen las características básicas de una base de datos relacional.

Sistema Gestor de Base de Datos (SGBD) o Relational Database Management System (RDBMS): programa que permite al usuario gestionar una base de datos, determina la forma en que los datos son guardados, mantenidos y extraídos. La gran mayoría están basados en el lenguaje SQL. Algunos de los gestores más extendidos son Oracle, el DB2 de IBM y el SQL Server de Microsoft.

Teorema CAP: enunciado por Eric Brewer, establece que un sistema distribuido no puede cumplir simultáneamente las tres propiedades CAP: Consistencia, Disponibiliad, Tolerante a particiones. Tiene tres combinaciones: CA, CP y AP.

Tolerante a particiones (CAP): si una parte de la base de datos deja de estar disponible, las partes restantes no se verán afectadas.

8. Webgrafía y bibliografía

En esta sección se organizan las referencias del material utilizado a lo largo de este trabajo.

Acuerdo de licencia de la versión Oracle XE 11g r2 (consultar qué funciones incluye o no): http://docs.oracle.com/cd/E17781_01/license.112/e18068/toc.htm#XELIC116

Anindya Das: https://www.youtube.com/watch?v=9IPS7uelbxg&list=PLUeF6A_iPYw3yE1VwmiF-tZJFJUZINgim

https://www.youtube.com/watch?v=MCOhtc_ubqc

Bases de datos NoSQL: http://blog.knuthaugen.no/2010/03/a-brief-history-of-nosql.html; https://www.3pillarglobal.com/insights/short-history-databases-rdbms-nosql-beyond; http://www.ijecbs.com/July2013/3.pdf.

Blog del experto en OracleDB Jeff Smith: http://www.thatjeffsmith.com/

Blog oficial de Oracle para resolución de dudas - Ask Tom: https://asktom.oracle.com

Blog oficial de Oracle para el Optimizer: https://blogs.oracle.com/optimizer/

Complete Oracle DB Documentation: https://docs.oracle.com/en/database/database.html

Database Trends and Applications: http://www.dbta.com/Editorial/Trends-and-Applications/Best-Database-Administration-Solution-112602.aspx

Descripción de las estadísticas disponibles en las vistas dinámicas: https://docs.oracle.com/cd/B28359_01/server.111/b28320/stats.htm#g1194481

Diagnostic Tools Overview Oracle: https://blogs.oracle.com/sql/entry/how_to_create_an_execution

Documentación SQL Developer: http://www.oracle.com/technetwork/developer-tools/sql-developer/documentation/index.html

Guía de usuarios de Toad: http://documents.software.dell.com/toad-for-oracle/12.9/user-guide

MariaDB: https://mariadb.com/

Navicat para Oracle: https://www.navicat.com/es/download/navicat-for-oracle

Novedades Oracle 11g: http://www.oracle.com/technetwork/articles/sql/11g-sqlplanmanagement-101938.html

Oracle Explain Plan Fundamentals: https://www.youtube.com/watch?v=9IPS7uelbxg

Oracle SQL Developer's Tuning: https://www.youtube.com/watch?v=rly5mRgQqD0

Oracle Autotrace Functionality: http://betteratoracle.com/posts/10-using-autotrace

Oracle execution plans: https://blogs.oracle.com/sql/entry/how_to_create_an_execution

Oracle New PL/SQL Packages in Oracle 11g R2: https://docs.oracle.com/cd/E11882_01/appdev.112/e40758/whatsnew.htm#ARPLS000

Oracle DB Online Documentation 11g Release 2 > DB Administration: https://docs.oracle.com/cd/E11882_01/nav/portal_4.htm

Pierrotti Checklist for Heuristic Evaluation: http://eitidaten.fh-pforzheim.de/daten/mitarbeiter/blankenbach/vorlesungen/GUI/Heuristic_Evaluation_Checklist_stcsig_org.pdf

PostgreSQL: https://www.postgresql.org/

Privilegios requeridos por Toad: http://documents.software.dell.com/toad-for-oracle/12.9/user-guide/required-privileges/required-oracle-privileges

SQL Developer User's Guide: https://docs.oracle.com/cd/E11882_01/doc.112/e12152/toc.htm

TKPROF: https://docs.oracle.com/cd/B28359_01/server.111/b28274/sqltrace.htm

TOAD Orafaq: http://www.orafaq.com/wiki/TOAD_Product_Review

TOAD 12.9 Release Notes: http://documents.software.dell.com/toad-for-oracle/12.9/release-notes/system-requirements

Bibliografía

Bolaños, V. (2.015, Marzo 9). *RTVE*. Retrieved Septiembre 3, 2.016, from http://www.rtve.es/noticias/20150309/nueva-ley-patentes-preve-proteger-software-supuesto-rechazado-ue-2005/1111540.shtml

Cornell University, World Economic Forum, INSEAD. (2.015). *The Global Information Technology Report 2015.*

Expansión. (2.015, Diciembre 23). *Expansión.* Retrieved Mayo 2, 2.016, from http://www.expansion.com/empresas/tecnologia/2015/12/23/5679a62be2704e523c8b4634.html

George, D. S. (2013). NoSQL - NotOnly SQL. *International Journal of Enterprise Computing and Business Systems*.

INE España. (n.d.). *Instituto Nacional de Estadística*. Retrieved Mayo 2, 2.016, from http://www.ine.es/jaxi/menu.do?type=pcaxis&path=%2Ft14%2Fp197%2Fe01&file=inebase&L=0

Nielsen Norman Group. (n.d.). Retrieved Agosto 25, 2.016, from https://www.nngroup.com/articles/ten-usability-heuristics/

OEPM. (n.d.). *Oficina Española de Patentes y Marcas*. Retrieved Septiembre 3, 2.016, from http://www.oepm.es/cs/OEPMSite/contenidos/Folletos/FOLLETO_3_PATENTAR_SOFTWARE/017-12_EPO_software_web.html

Oracle Community. (n.d.). Retrieved Septiembre 2.016, from https://community.oracle.com/community/database/developer-tools/sql_developer

Oracle Docs. (n.d.). Retrieved Febrero 16, 2.016, from http://docs.oracle.com/cd/B28359_01/server.111/b28274/optimops.htm

Pierrotti Nielsen Heuristics. (n.d.). Retrieved Agosto 14, 2.016, from http://eitidaten.fh-pforzheim.de/daten/mitarbeiter/blankenbach/vorlesungen/GUI/Heuristic_Evaluation_Checklist_stcsig_org.pdf

PwC. (2.016). *19th Annual Global CEO Survey: Technology industry key findings.* Retrieved Mayo 2, 2.016, from http://www.pwc.es/es/publicaciones/19-encuesta-mundial-ceos/assets/ceos-tecnologia.pdf

Toad World Official Page. (n.d.). Retrieved Agosto 2.016, from http://www.toadworld.com/

US Bureau of Labor Statistics. (2012). *Industry employment and output projections to 2020.*

US Bureau of Labor Statistics. (n.d.). *Bureau of Labor Statistics*. Retrieved Abril 27, 2.016, from http://www.bls.gov/iag/tgs/iag51.htm

9. Anexos

A continuación se presentan los dos anexos del estudio. En primer lugar, las tablas de mayor tamaño que no han sido incluidas previamente en la sección 4. En segundo lugar, el modelo de encuesta utilizado en el estudio.

9.1 Manual de Instalación de las Herramientas.

Instalación de Oracle XE 11g r2:

1. Accedemos con nuestra cuenta personal a la sección de descargas de Oracle Database Edición Express 11g. Descargamos el archivo correspondiente a nuestro sistema operativo (en mi caso, para Windows 64bits). Al ser la edición Express, su peso (317MB aprox) es mucho menor que el de la edición Enterprise (que supera los 2,5GB).

http://www.oracle.com/technetwork/database/database-technologies/express-edition/downloads/index.html

En esa misma página, si navegamos a la pestaña "Documentation", encontraremos toda la documentación oficial de Oracle para esta versión:

http://docs.oracle.com/cd/E17781_01/index.htm

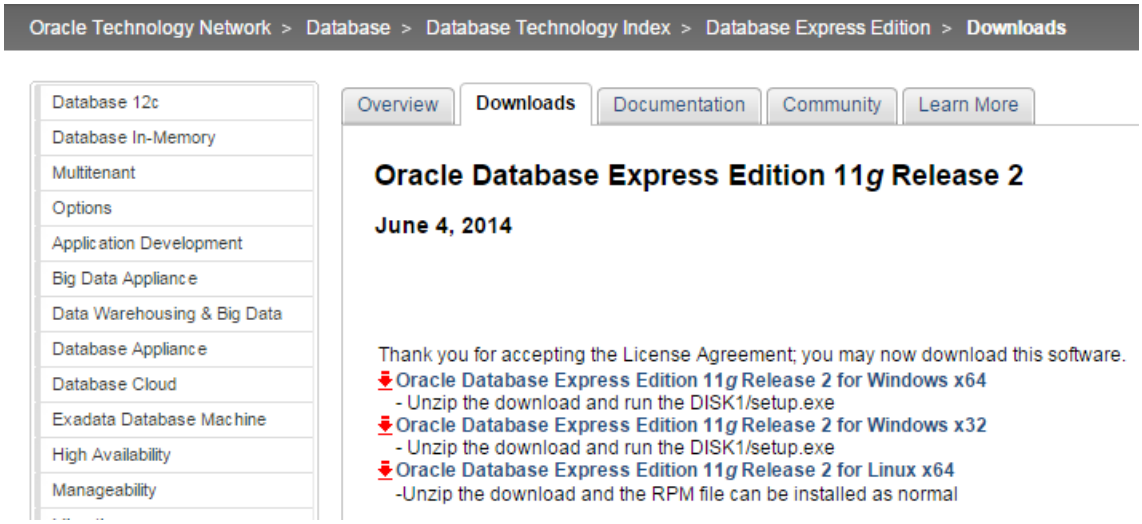


Ilustración 37: Instalación Oracle XE 11g r2 (I)

2. Una vez descargado el archive en una carpeta local, lo descomprimimos.

3. Antes de proceder a su instalación, hay que asegurarse de que no existe previamente ninguna otra Oracle DB XE u otra base de datos con el SID "XE" en el sistema en que vamos a instalarlo. En caso de haber una versión ya instalada, lo más recomendable es exportar los datos y luego volver a

importarlos tras la nueva instalación. (Para más información, ver: http://docs.oracle.com/cd/E17781_01/install.112/e18803/toc.htm#XEINW136)

4. Seguiremos el manual:

   http://docs.oracle.com/cd/E17781_01/install.112/e18803/toc.htm#XEINW124

5. Variable de entorno "ORACLE_HOME". Comprobar que no hay ya establecida una variable de entorno en el sistema ya asociada (lo podemos comprobar en la consola de Windows cmd con el comando "echo %ORACLE_HOME%", que en el caso de que esté establecida devolverá su ruta). Si esto sucede, hay que eliminarla.

6. Una vez descomprimido el archivo, hacemos clic en el archivo "setup.exe" y lo ejecutamos. Después de aceptar las condiciones, nos pide elegir la ruta de destino. Por defecto usará "C:\oraclexe\", aunque podemos seleccionar otra si así lo queremos (yo voy a dejarlo en la propuesta).

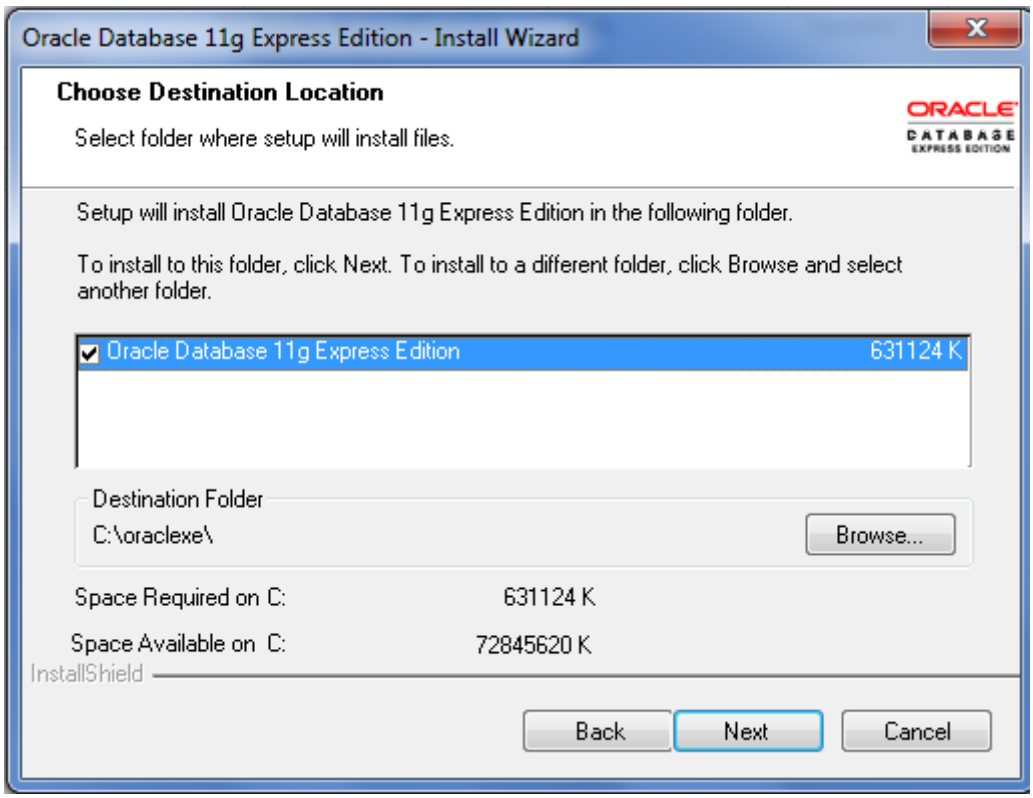


Ilustración 38: Instalación Oracle XE 11g r2 (II)

7. Especificamos una contraseña que vaya a servir para las cuentas por defecto de SYS y SYSTEM:

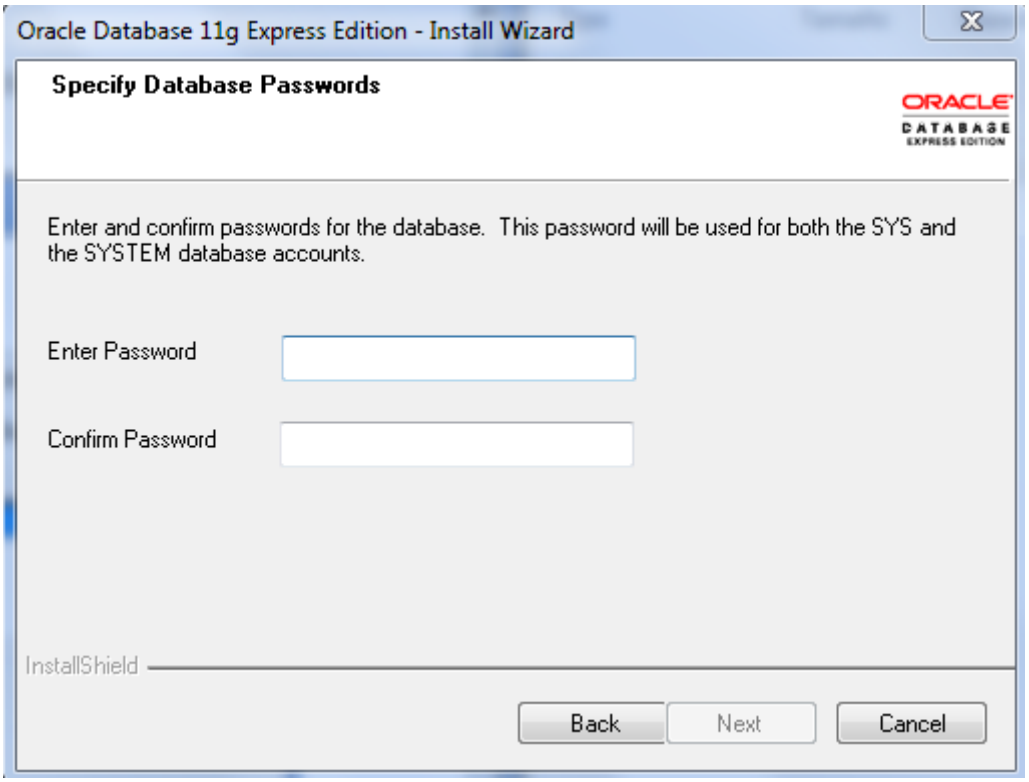


Ilustración 39: Instalación Oracle XE 11g r2 (III)

**Importante: la contraseña para las cuentas de usuario "INTERNAL" y "ADMIN" de Oracle Application XE será la misma que la especificada para las cuentas de administración "SYS" y "SYSTEM".

8. Antes de confirmar la instalación, nos saldrá un resumen de la configuración:

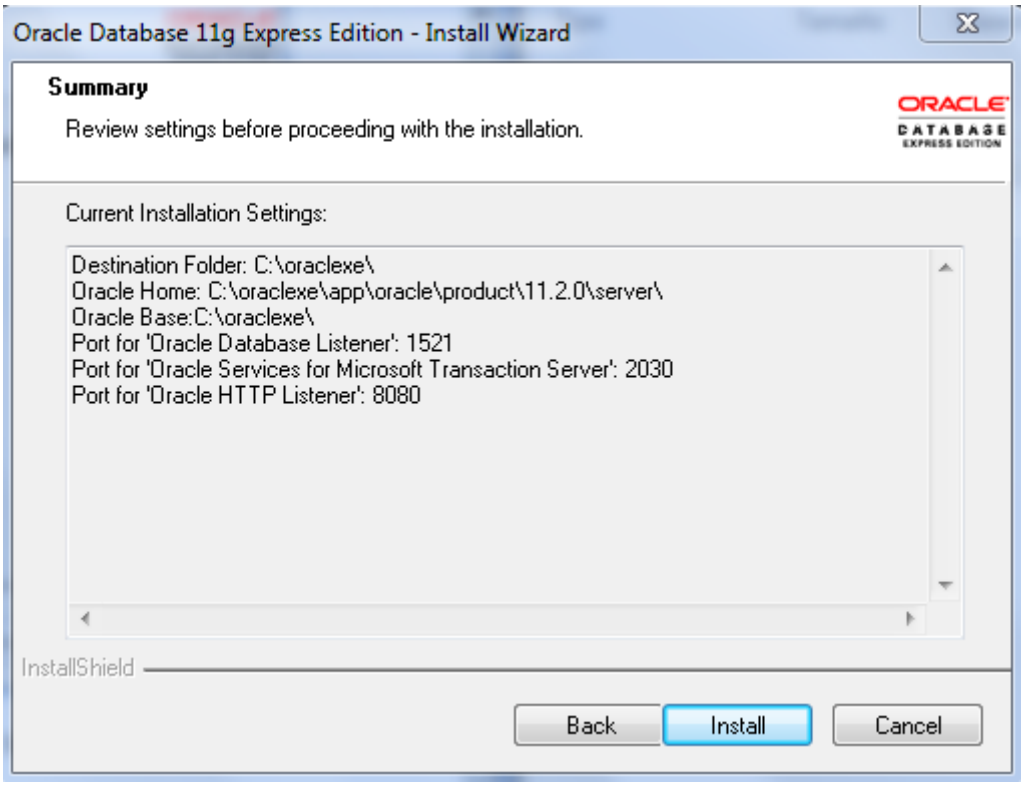


Ilustración 40: Instalación Oracle XE 11g r2 (IV)

Estos puertos asignados por defecto son:

- 1521: Oracle DB listener.
- 2030: Oracle Services for Microsoft Transaction Server.
- 8080: Puerto HTTP para la interfaz gráfica de Oracle DB XE.

En el caso de que los números de puerto no están siendo utilizados, la instalación los asignará automáticamente, pero si están en uso se pedirá al usuario que asigne unos.

9. Cuando haya terminado de instalarse con éxito aparecerá el siguiente mensaje:

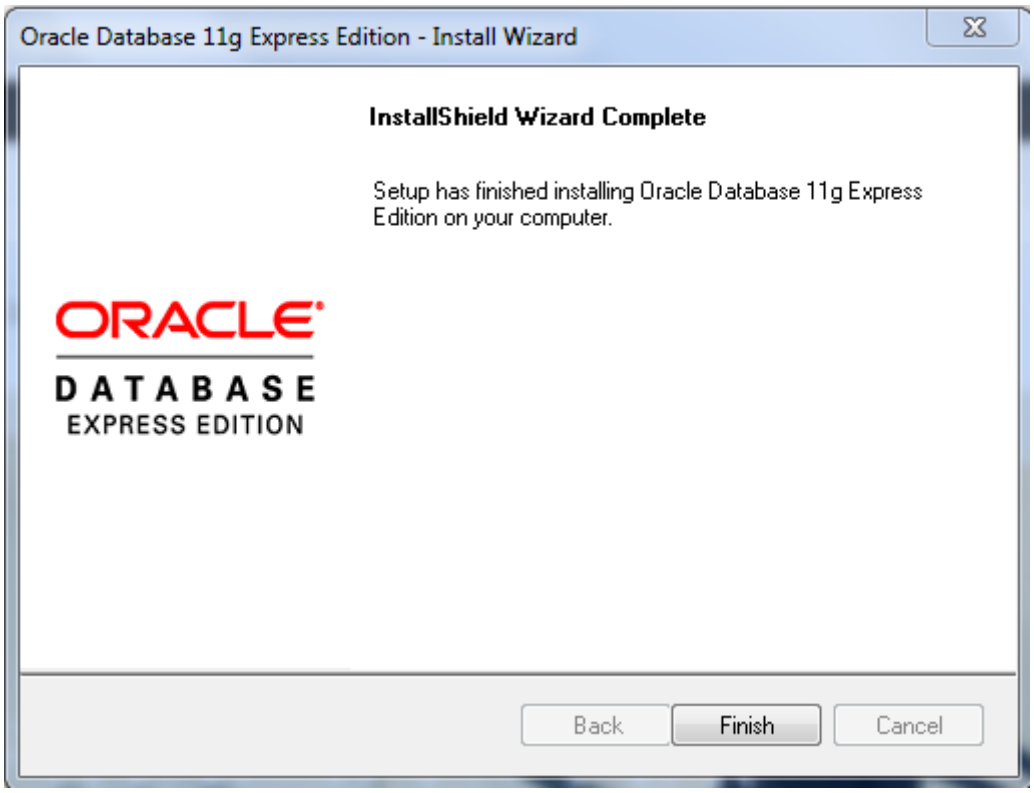


Ilustración 41: Instalación Oracle XE 11g r2 (V)

10. Para comenzar a usarlo, vamos a Inicio > Todos los programas > Oracle Database 11g Express Edition > Run SQL Command Line:

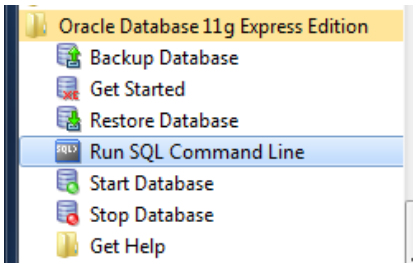


Ilustración 42: Instalación Oracle XE 11g r2 (VI)

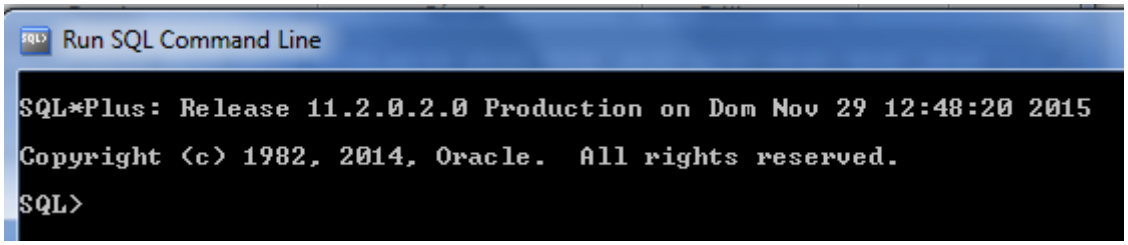


Ilustración 43: Instalación Oracle XE 11g r2 (VII)

Instalación de SQL Developer:

1. Descargamos SQL developer, lo descomprimimos y hacemos clic en "sqldeveloper.exe" de http://www.oracle.com/technetwork/developer-tools/sql-developer/downloads/index.html. Pantalla inicial:

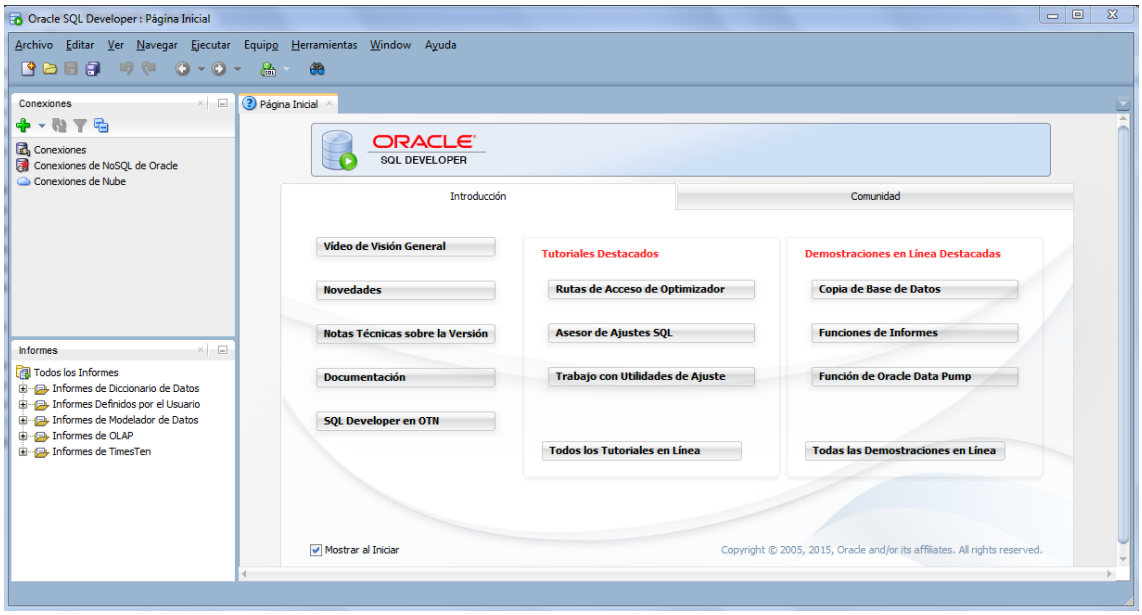


Ilustración 44: Instalación SQL Developer (I)

2. Para crear una conexión, primero hay que crear un usuario con lo suficientes privilegios a través de Oracle DB XE. https://docs.oracle.com/cd/E17781_01/admin.112/e18585/toc.htm#XEGSG126

   Hay que tener en cuenta que eso se hace desde la consola de Oracle XE, logándose primero como usuario system.

3. Este usuario por ejemplo será 'raquelgf'. A la hora de crear una nueva conexión:

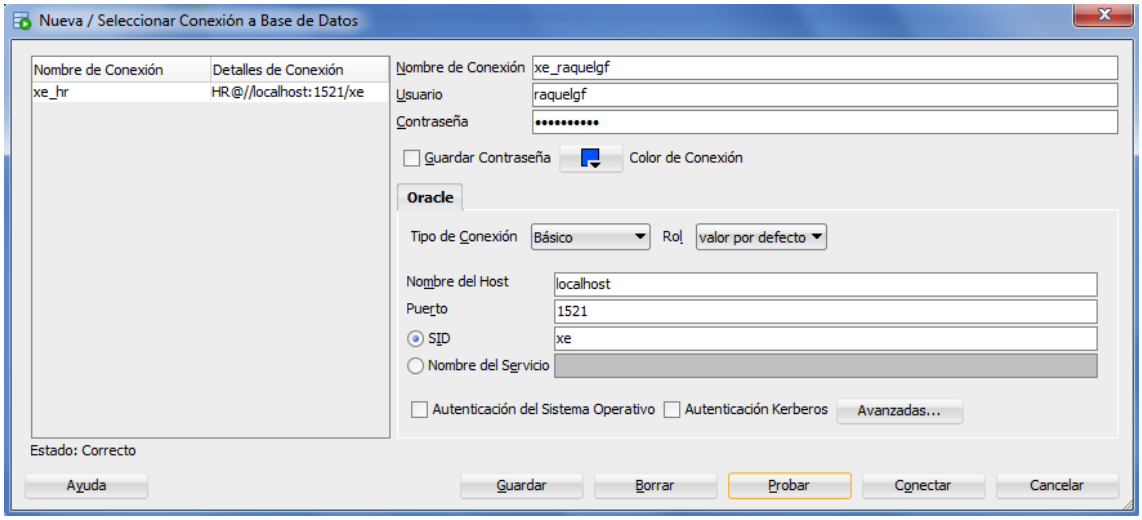


Ilustración 45: Instalación SQL Developer (II)

**Tipos de roles que pueden seleccionarse: Valor por defecto, SYSDBA, SYSBACKUP, SYSDG, SYSKM y SYSASM.

4. Otra opción es desbloquear el esquema por defecto “HR”, que ya viene con tablas e índices creados. Esta es la opción escogida para el desarrollo del proyecto.

Puede hacerse vía consola o usando SQL Developer. Las acciones son similares (primero desbloquear la cuenta y luego asignarle una contraseña), por lo que la decisión de utilizar un medio u otro depende de los criterios de rapidez (consola) y visibilidad gráfica de las acciones (SQL Developer).

Una vez desbloqueado el usuario HR, es posible (y necesario para esta práctica) crear una conexión asociada a él. Dentro de esta conexión podrán verse en SQL Developer los objetos ya incluidos por defecto.

Para obtener información detallada sobre el desbloqueo de este usuario, consultar: http://docs.oracle.com/cd/E17781_01/admin.112/e18585/toc.htm#XEGSG102.

5. Para ver los usuarios y los permisos que tienen:

   SQL> select grantee, granted_role from dba_role_privs;

6. Conocer los privilegios del usuario conectado:

```
SQL> connect raquelgf/tequiero92
Connected.
SQL> select * from user_role_privs;

no rows selected

SQL> select * from user_sys_privs;

USERNAME                       PRIVILEGE                                ADM
------------------------------ ---------------------------------------- ---
RAQUELGF                       CREATE ROLE                              NO
RAQUELGF                       CREATE SYNONYM                           NO
RAQUELGF                       CREATE TYPE                              NO
RAQUELGF                       CREATE DATABASE LINK                     NO
RAQUELGF                       ALTER SESSION                            NO
RAQUELGF                       CREATE MATERIALIZED VIEW                 NO
RAQUELGF                       CREATE TABLE                             NO
RAQUELGF                       UNLIMITED TABLESPACE                     NO
RAQUELGF                       CREATE SESSION                           NO
RAQUELGF                       CREATE PROCEDURE                         NO
RAQUELGF                       CREATE TRIGGER                           NO

USERNAME                       PRIVILEGE                                ADM
------------------------------ ---------------------------------------- ---
RAQUELGF                       CREATE SEQUENCE                          NO
RAQUELGF                       CREATE VIEW                              NO
RAQUELGF                       CREATE PUBLIC SYNONYM                    NO

14 rows selected.
```

Ilustración 46: Instalación SQL Developer (III)

7. Rastreo automático:

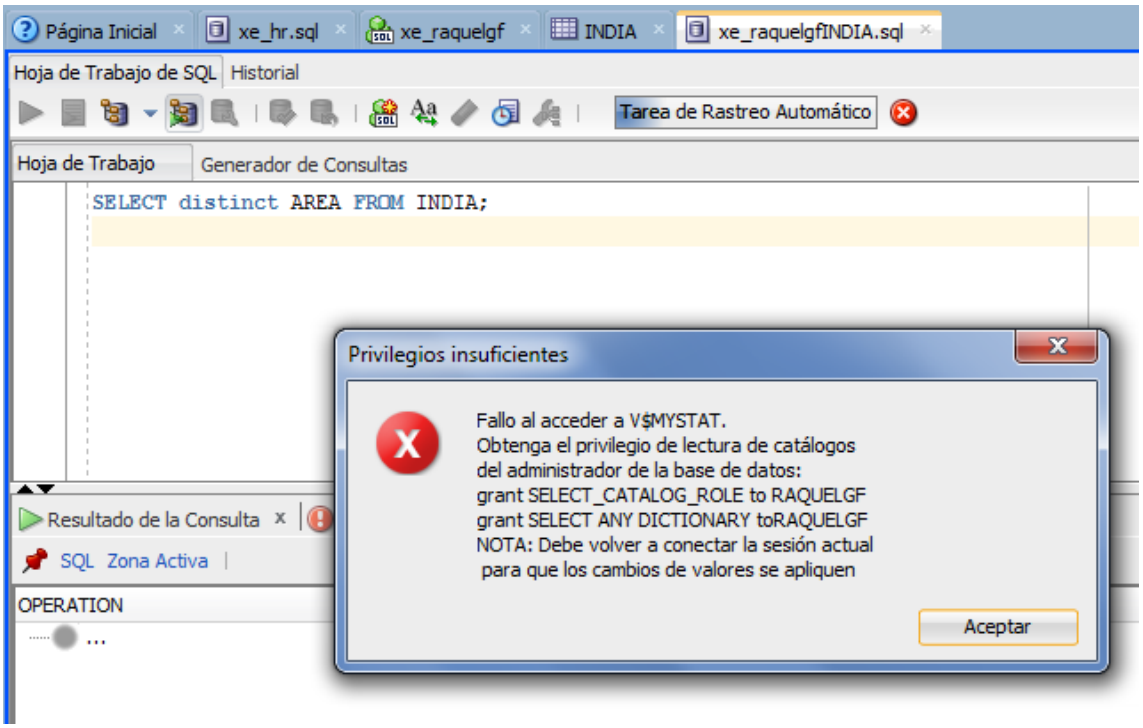


Ilustración 47: Instalación SQL Developer (IV)

Instalación de TOAD (prueba gratuita de 30 días):

Hay una introducción general con presentación de los productos en la web https://software.dell.com/mx-es/products/toad-for-oracle/. De todos ellos, se ha probado la edición Xpert, que es la que incluye las funcionalidades de optimización SQL (https://software.dell.com/mx-es/products/toad-for-oracle/xpert-edition.aspx). Para descargarlo se habrá tenido que facilitar un correo electrónico, al que se envía la confirmación del pedido junto con una license key y un mensaje. Estos dos datos serán solicitados por el instalador para continuar el proceso.

Una vez dentro, para añadir una conexión, se puede tomar como referencia las conexiones creadas en SQL Developer. Por ejemplo, la conexión para el usuario HR sería:

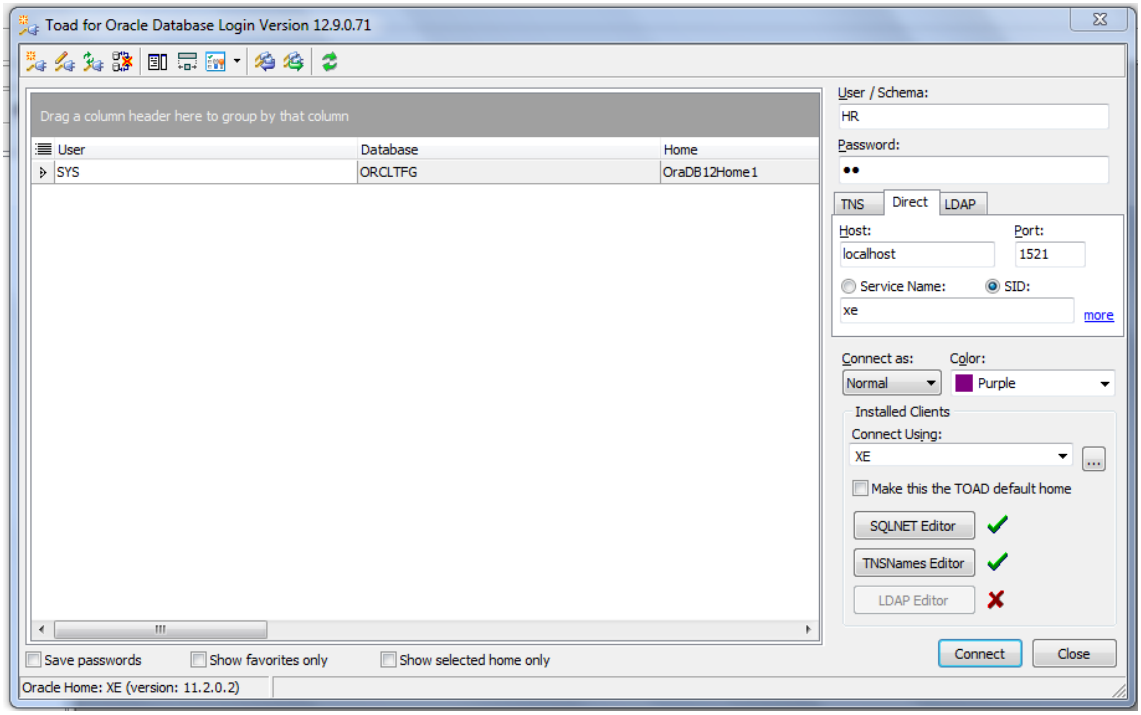


Ilustración 48: Instalación TOAD (I)

Basado en:

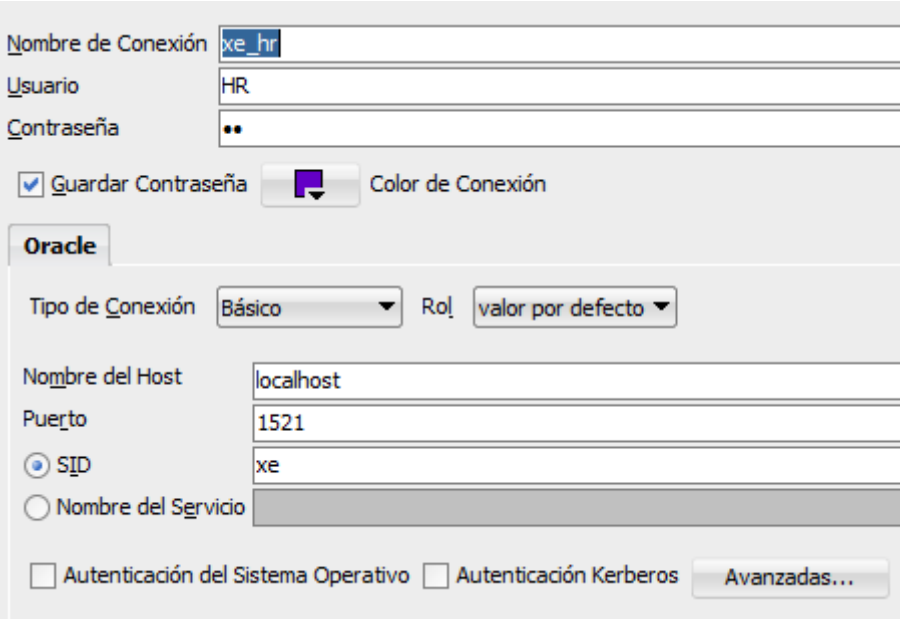


Ilustración 49: Instalación TOAD (II)

Para acceder al contenido de ese esquema, basta con pulsar el botón del menú "Database" > "Schema Browser".

9.2 Comandos de ejecución útiles

Parámetros de inicialización:

Comprobar el valor de los parámetros: ‘`show parameters`’.

Comprobar si un parámetro de la lista ‘`show parameters`’ o de la vista `v$parameter` es modificable: `select name, isses_modifiable, issys_modifiable from v$parameter where name='pga_aggregate_target';`

Habilitar la función Autotrace en SQL Developer. En caso de que el DBA no quiera otorgar "grant select_catalog_role to HR" o "grant select any dictionary to HR", que son los privilegios que necesita el usuario HR para acceder a v$mystat y usar SQLDevAutotrace, existe una alternativa (que debe ser ejecutada por el DBA y no surtirá efecto hasta que se reinicie la sesión):

```
drop role plustrace;
create role plustrace;
grant select on v_$sesstat to plustrace;
grant select on v_$statname to plustrace;
grant select on v_$mystat to plustrace;
grant select on v_$session to plustrace;
grant select on v_$sql_plan to plustrace;
grant select on v_$sql_plan_statistics to plustrace;
grant select on v_$sql to plustrace;
grant plustrace to dba with admin option;
grant plustrace to HR;
```

Activar la función DBMS_MONITOR.SESSION_TRACE:

El método session_trace_enable consta de cinco parámetros: session_id, serial_num, waits, binds y plan_stat. El alcance en el caso de este estudio es la propia sesión del usuario que llame al método, añadiendo información de los waits y binds, por lo que el comando sería: `execute dbms_monitor.session_trace_enable(null,null,waits => TRUE, binds => TRUE);`.

Para desactivarlo: `execute dbms_monitor.session_trace_disable;`.

Generar el informe TKPROF: Primero se tiene que haber activado la función session_trace_enable (‘session’ suponiendo que sólo se quiera aplicar para la sesión actual).

```
execute dbms_monitor.session_trace_enable(null,null,waits => TRUE, binds
=> TRUE);

alter session set tracefile_identifier='c1_pA';
```

[Se ejecutan las consultas que se quieran analizar]

```
execute dbms_monitor.session_trace_disable;
```

`host //` Retroceder con cd.. hasta llegar a C:\ y luego acceder a " cd `C:\oraclexe\app\oracle\diag\rdbms\xe\xe\trace>tkprof (archivo tracefile de origen).trc (archivo tkprof de destino).prf`". *Atención, el nombre del tracefile contiene el identificador, pero no es el nombre completo, por lo que hay que acceder a la ruta user_dump_dest (desde la que se lanza el tkprof) y comprobar el nombre real del archivo.

Para volver al modo SQL: `exit`

Obtener el SQL_ID de una consulta: Se recomienda insertar en la consulta de la que se quiera obtener el sql_id un comentario que la identifique, y luego utilizarlo como parámetro de búsqueda. Aparecerán dos resultados, el correcto es el primero, porque el segundo corresponde al de la consulta ejecutada para buscar el sql_id:

```
select sql_id, substr(sql_text,1,200) sql_text from v$sql where sql_text like '%(comentario)%';
```

Utilizar variables bind:

Se crea como "`variable dept_name VARCHAR2(30)`".

Se declara el valor "`exec :dept_name := 'Marketing_EMEA'`".

Se aplica la sentencia haciendo referencia a la variable bind: "`select distinct count(employees.employee_id),departments.department_id from departments,employees where departments.department_id=employees.department_id and departments.department_name = :dept_name group by departments.department_id order by departments.department_id;`".

Se consulta las variables bind creadas en la sesión actual con "`variable`" (sin argumentos).

Función dbms_xplan.display_cursor:

`grant execute on dbms_xplan to hr;` // concedido por sys as sysdba a hr, el usuario que lanzará el procedimiento

`alter session set statistics_level='ALL'` // para poder ejecutar la función display_cursor

Lanzar la consulta con '`explain plan for`'

Lanzar la consulta sin '`explain plan for`' // De aquí interesa obtener el sql_id

Obtener el `sql_id` de la consulta en sí.

`select * from table(dbms_xplan.display_cursor('f36jqafhgtbrx',0,'all allstats'));` //Muestra el plan de ejecución, el valor de A-rows y E-rows, y las estadísticas iostats y memstats si cumplen los requisitos.

Obtener el estado de las sesiones actuales, identificadas por su sid: `select sid, sql_id, status, module from v$session;`

Obtener los ID de las estadísticas relacionadas con SQL (Class = 64): `select statistic#, name from v$statname where class=64;`

Extraer AutotraceSQLDev, XplanSQLDev y Tracefile + TKPROF en SQL Developer:

Si no se ha hecho antes, desde una conexión *sys as sysdba*:

```
grant execute on dbms_xplan to hr;

grant execute on dbms_monitor to hr;
```

Desde el usuario HR:

```
alter session set statistics_level='ALL'

-------- Lanzar la consulta con 'explain plan for'

explain plan FOR SELECT /* com1 */ DISTINCT COUNT (employees.employee_id), departments.department_id FROM departments, employees WHERE departments.department_id = employees.department_id

AND departments.department_name = 'Marketing_Asia'

group by departments.department_id

order by departments.department_id;

-------------------------------------------

execute dbms_monitor.session_trace_enable(null,null,waits => TRUE, binds => TRUE);

alter session set tracefile_identifier='trace_com1';

--------lanzo la consulta sin 'explain plan', sacar de aquí el sql_id

SELECT /* com1 */ DISTINCT COUNT (employees.employee_id), departments.department_id FROM departments, employees WHERE departments.department_id = employees.department_id

AND departments.department_name = 'Marketing_Asia'

group by departments.department_id

order by departments.department_id;

-------------------------------------------

select sql_id, substr(sql_text,1,200) sql_text from v$sql where sql_text like '%com1%';
```

```
select * from table(dbms_xplan.display_cursor('sql_id_obtenido',0,'all
allstats last'));

execute dbms_monitor.session_trace_disable;
```

Desde la cmd de Windows, retroceder hasta el cd C:\ y luego ir hasta "C:\oraclexe\app\oracle\diag\rdbms\xe\xe\trace". Ahí, lanzar:

```
tkprof origen_tracefile.trc destino_TKPROF.prf
```

9.3 Alternativas de las consultas TOAD

Este anexo está dedicado a presentar las diferentes alternativas optimizadas de cada consulta mencionadas anteriormente en el apartado 4.6.

Consulta 1:

- Mejora c1_p1_toad:

```
SELECT /*c1_p1_toad*/ Q1.DEPARTMENT_ID,
         DEPARTMENT_NAME,
         REGION_NAME,
         Q3.MANAGER_ID,
         SUBORDINADOS
    FROM (SELECT DEPARTMENT_ID, DEPARTMENT_NAME, REGION_NAME
            FROM HR.DEPARTMENTS,
                 HR.LOCATIONS,
                 HR.COUNTRIES,
                 HR.REGIONS
           WHERE     DEPARTMENTS.LOCATION_ID = LOCATIONS.LOCATION_ID
                 AND LOCATIONS.COUNTRY_ID = COUNTRIES.COUNTRY_ID
                 AND COUNTRIES.REGION_ID = REGIONS.REGION_ID) Q1,
         (SELECT DISTINCT DEPARTMENTS.DEPARTMENT_ID,
DEPARTMENTS.MANAGER_ID
            FROM HR.DEPARTMENTS, HR.EMPLOYEES, HR.JOBS
           WHERE     JOB_TITLE LIKE '%Finance%'
                 AND DEPARTMENTS.MANAGER_ID = EMPLOYEE_ID
                 AND EMPLOYEES.JOB_ID = JOBS.JOB_ID) Q2,
         (SELECT EMPLOYEES.MANAGER_ID, COUNT (*) SUBORDINADOS
             FROM HR.DEPARTMENTS, HR.EMPLOYEES
            WHERE     HIRE_DATE > '01/01/2000'
                  AND DEPARTMENTS.MANAGER_ID = EMPLOYEES.MANAGER_ID
          GROUP BY EMPLOYEES.MANAGER_ID) Q3
   WHERE Q1.DEPARTMENT_ID = Q2.DEPARTMENT_ID AND Q2.MANAGER_ID =
Q3.MANAGER_ID
ORDER BY Q3.MANAGER_ID ASC;
```

- Mejora c1_p2_toad:

```
SELECT /*c1_p2_toad*/ Q1.DEPARTMENT_ID,
         DEPARTMENT_NAME,
         REGION_NAME,
         Q3.MANAGER_ID,
         SUBORDINADOS
    FROM (SELECT DEPARTMENT_ID, DEPARTMENT_NAME, REGION_NAME
```

```
            FROM HR.DEPARTMENTS
                 INNER JOIN HR.LOCATIONS
                    ON (DEPARTMENTS.LOCATION_ID =
LOCATIONS.LOCATION_ID)
                 INNER JOIN HR.COUNTRIES
                    ON (LOCATIONS.COUNTRY_ID = COUNTRIES.COUNTRY_ID)
                 INNER JOIN HR.REGIONS
                    ON (COUNTRIES.REGION_ID = REGIONS.REGION_ID)) Q1
         INNER JOIN
         (SELECT DISTINCT DEPARTMENTS.DEPARTMENT_ID,
DEPARTMENTS.MANAGER_ID
            FROM HR.DEPARTMENTS
                 INNER JOIN HR.EMPLOYEES
                    ON (DEPARTMENTS.MANAGER_ID = EMPLOYEE_ID)
                 INNER JOIN HR.JOBS ON (EMPLOYEES.JOB_ID =
JOBS.JOB_ID)
           WHERE JOB_TITLE LIKE '%Finance%') Q2
            ON (Q1.DEPARTMENT_ID = Q2.DEPARTMENT_ID)
         INNER JOIN
         (  SELECT EMPLOYEES.MANAGER_ID, COUNT (*) SUBORDINADOS
              FROM HR.DEPARTMENTS
                   INNER JOIN HR.EMPLOYEES
                      ON (DEPARTMENTS.MANAGER_ID =
EMPLOYEES.MANAGER_ID)
             WHERE HIRE_DATE > '01/01/2000'
          GROUP BY EMPLOYEES.MANAGER_ID) Q3
            ON (Q2.MANAGER_ID = Q3.MANAGER_ID)
ORDER BY Q3.MANAGER_ID ASC;
```

- Mejoras c1_p3a_toad y c1_p7a_toad:

```
select /*+ USE_HASH(Q3,Q2) ORDERED */ q1.department_id,
       q1.department_name,
       q1.region_name,
       q3.manager_id,
       q3.subordinados
  FROM (select DEPARTMENTS1.department_id,
               DEPARTMENTS1.department_name,
               regions.region_name
          FROM hr.departments DEPARTMENTS1
               INNER JOIN hr.locations
                  ON DEPARTMENTS1.location_id = locations.location_id
               INNER JOIN hr.countries
                  ON locations.country_id = countries.country_id
               INNER JOIN hr.regions
                  ON countries.region_id = regions.region_id) q1
       INNER JOIN (select distinct DEPARTMENTS2.department_id,
                                   DEPARTMENTS2.manager_id
                     FROM hr.departments DEPARTMENTS2
                          INNER JOIN hr.employees EMPLOYEES1
                             ON DEPARTMENTS2.manager_id =
EMPLOYEES1.employee_id
                          INNER JOIN hr.jobs
                             ON EMPLOYEES1.job_id = jobs.job_id
                    WHERE jobs.job_title like '%Finance%') q2
          ON q1.department_id = q2.department_id
       INNER JOIN (select EMPLOYEES2.manager_id,
                          count(*) as subordinados
                     FROM hr.departments DEPARTMENTS3
                          INNER JOIN hr.employees EMPLOYEES2
```

```
                              ON DEPARTMENTS3.manager_id =
EMPLOYEES2.manager_id
                    WHERE EMPLOYEES2.hire_date > '01/01/2000'
                    group by EMPLOYEES2.manager_id) q3
          ON q2.manager_id = q3.manager_id
 order by q3.manager_id asc;
```

- Mejoras c1_p3b_toad, c1_p6b_toad y c1_p7b_toad:

```
select q1.department_id,
       q1.department_name,
       q1.region_name,
       q3.manager_id,
       q3.subordinados
  FROM (select DEPARTMENTS1.department_id,
               DEPARTMENTS1.department_name,
               regions.region_name
          FROM hr.departments DEPARTMENTS1
               INNER JOIN hr.regions
                  ON 1 = 1
               INNER JOIN hr.locations
                  ON DEPARTMENTS1.location_id = locations.location_id
+ 0
               INNER JOIN hr.countries
                  ON regions.region_id = countries.region_id + 0
                     AND locations.country_id = countries.country_id
|| '') q1
       INNER JOIN (select /*+ USE_NL(JOBS) */ distinct
DEPARTMENTS2.department_id,
                          DEPARTMENTS2.manager_id
                     FROM hr.departments DEPARTMENTS2
                          INNER JOIN hr.employees EMPLOYEES1
                             ON DEPARTMENTS2.manager_id =
EMPLOYEES1.employee_id
                          INNER JOIN hr.jobs
                             ON EMPLOYEES1.job_id = jobs.job_id
                    WHERE jobs.job_title like '%Finance%') q2
          ON q1.department_id = q2.department_id
       INNER JOIN (select EMPLOYEES2.manager_id,
                          count(*) as subordinados
                     FROM hr.departments DEPARTMENTS3
                          INNER JOIN hr.employees EMPLOYEES2
                             ON DEPARTMENTS3.manager_id =
EMPLOYEES2.manager_id
                    WHERE EMPLOYEES2.hire_date > '01/01/2000'
                    group by EMPLOYEES2.manager_id) q3
          ON q2.manager_id = q3.manager_id
 order by q3.manager_id asc;
```

- Mejora c1_p5a_toad:

```
select /*+ USE_HASH(Q3,Q2) ORDERED */ q1.department_id,
       q1.department_name,
       q1.region_name,
       q3.manager_id,
       q3.subordinados
  FROM (select DEPARTMENTS1.department_id,
               DEPARTMENTS1.department_name,
               regions.region_name
          FROM hr.departments DEPARTMENTS1
               INNER JOIN hr.locations
```

```
                  ON DEPARTMENTS1.location_id = locations.location_id
+ 0
               INNER JOIN hr.countries
                  ON locations.country_id = countries.country_id || ''
               INNER JOIN hr.regions
                  ON countries.region_id = regions.region_id) q1
       INNER JOIN (select distinct DEPARTMENTS2.department_id,
                           DEPARTMENTS2.manager_id
                      FROM hr.departments DEPARTMENTS2
                           INNER JOIN hr.employees EMPLOYEES1
                              ON DEPARTMENTS2.manager_id =
EMPLOYEES1.employee_id
                           INNER JOIN hr.jobs
                              ON EMPLOYEES1.job_id = jobs.job_id
                     WHERE jobs.job_title like '%Finance%') q2
          ON q1.department_id = q2.department_id
       INNER JOIN (select EMPLOYEES2.manager_id,
                           count(*) as subordinados
                      FROM hr.departments DEPARTMENTS3
                           INNER JOIN hr.employees EMPLOYEES2
                              ON DEPARTMENTS3.manager_id =
EMPLOYEES2.manager_id
                     WHERE EMPLOYEES2.hire_date > '01/01/2000'
                     group by EMPLOYEES2.manager_id) q3
          ON q2.manager_id = q3.manager_id
 order by q3.manager_id asc;
```

- Mejoras c1_p5b_toad y c1_p8b_toad:

```
select q1.department_id,
       q1.department_name,
       q1.region_name,
       q3.manager_id,
       q3.subordinados
  FROM (select DEPARTMENTS1.department_id,
               DEPARTMENTS1.department_name,
               regions.region_name
          FROM hr.departments DEPARTMENTS1
               INNER JOIN hr.regions
                  ON 1 = 1
               INNER JOIN hr.locations
                  ON DEPARTMENTS1.location_id = locations.location_id
+ 0
               INNER JOIN hr.countries
                  ON regions.region_id = countries.region_id + 0
                     AND locations.country_id = countries.country_id
|| '') q1
       INNER JOIN (select /*+ LEADING(JOBS) */ distinct
DEPARTMENTS2.department_id,
                           DEPARTMENTS2.manager_id
                      FROM hr.departments DEPARTMENTS2
                           INNER JOIN hr.employees EMPLOYEES1
                              ON DEPARTMENTS2.manager_id =
EMPLOYEES1.employee_id
                           INNER JOIN hr.jobs
                              ON EMPLOYEES1.job_id = jobs.job_id
                     WHERE jobs.job_title like '%Finance%') q2
          ON q1.department_id = q2.department_id
       INNER JOIN (select EMPLOYEES2.manager_id,
                           count(*) as subordinados
                      FROM hr.departments DEPARTMENTS3
```

```
                        INNER JOIN hr.employees EMPLOYEES2
                           ON DEPARTMENTS3.manager_id =
EMPLOYEES2.manager_id
                     WHERE EMPLOYEES2.hire_date > '01/01/2000'
                     group by EMPLOYEES2.manager_id) q3
          ON q2.manager_id = q3.manager_id
 order by q3.manager_id asc;
```

- Mejora c1_p6b_toad: es idéntica a la mejora c1_p3b_toad.
- Mejora c1_p7a_toad: La alternativa 29, que es la siguiente mejor a la original, es idéntica a la mejora c1_p3a_toad.
- Mejora c1_p7b_toad: es idéntica a la mejora c1_p3b_toad.
- Mejora c1_p8a_toad:

```
/* Alternativa 147 mejor tiempo (por detrás de la original)*/
 select /*+ NO_MERGE(Q1) */ q1.department_id,
       q1.department_name,
       q1.region_name,
       q3.manager_id,
       q3.subordinados
  FROM (select DEPARTMENTS1.department_id,
               DEPARTMENTS1.department_name,
               regions.region_name
          FROM hr.departments DEPARTMENTS1
               INNER JOIN hr.regions
                  ON 1 = 1
               INNER JOIN hr.locations
                  ON DEPARTMENTS1.location_id = locations.location_id
+ 0
               INNER JOIN hr.countries
                  ON regions.region_id = countries.region_id + 0
                     AND locations.country_id = countries.country_id
|| '') q1
       INNER JOIN (select distinct DEPARTMENTS2.department_id,
                           DEPARTMENTS2.manager_id
                      FROM hr.departments DEPARTMENTS2
                           INNER JOIN hr.employees EMPLOYEES1
                              ON DEPARTMENTS2.manager_id =
EMPLOYEES1.employee_id
                           INNER JOIN hr.jobs
                              ON EMPLOYEES1.job_id = jobs.job_id
                     WHERE jobs.job_title like '%Finance%') q2
          ON q1.department_id = q2.department_id
       INNER JOIN (select EMPLOYEES2.manager_id,
                           count(*) as subordinados
                      FROM hr.departments DEPARTMENTS3
                           INNER JOIN hr.employees EMPLOYEES2
                              ON DEPARTMENTS3.manager_id =
EMPLOYEES2.manager_id
                     WHERE EMPLOYEES2.hire_date > '01/01/2000'
                     group by EMPLOYEES2.manager_id) q3
          ON q2.manager_id = q3.manager_id
 order by q3.manager_id asc;
```

- Mejora c1_p8b_toad: es idéntica a la mejora c1_p5b_toad.

Consulta 2:

- Mejora c2_p2_toad: ANSI

```
SELECT REGION_NAME,
         DEPARTMENT_ID,
         Q2.JOB_ID,
         AVG_SAL_JOBID,
         MAX_SAL_JOBID,
         MAS_AVG,
         MIN_SAL_JOBID,
         MENOS_AVG,
         TOTAL_JOBID
    FROM (SELECT DISTINCT REGION_NAME,
DEPARTMENTS.DEPARTMENT_ID, JOBS.JOB_ID
            FROM HR.JOBS
                 INNER JOIN HR.EMPLOYEES ON (JOBS.JOB_ID =
EMPLOYEES.JOB_ID)
                 INNER JOIN HR.DEPARTMENTS
                    ON (EMPLOYEES.DEPARTMENT_ID =
DEPARTMENTS.DEPARTMENT_ID)
                 INNER JOIN HR.LOCATIONS
                    ON (DEPARTMENTS.LOCATION_ID =
LOCATIONS.LOCATION_ID)
                 INNER JOIN HR.COUNTRIES
                    ON (LOCATIONS.COUNTRY_ID =
COUNTRIES.COUNTRY_ID)
                 INNER JOIN HR.REGIONS
                    ON (COUNTRIES.REGION_ID = REGIONS.REGION_ID)
           WHERE JOB_TITLE LIKE '%Marketing%') Q1
         CROSS JOIN
         (  SELECT DISTINCT EMPLOYEES.JOB_ID, AVG (SALARY)
AVG_SAL_JOBID
              FROM HR.JOBS
                   INNER JOIN HR.EMPLOYEES ON (JOBS.JOB_ID =
EMPLOYEES.JOB_ID)
             WHERE JOB_TITLE LIKE '%Marketing%'
          GROUP BY EMPLOYEES.JOB_ID) Q2
         CROSS JOIN
         (  SELECT DISTINCT EMPLOYEES.JOB_ID, MAX (SALARY)
MAX_SAL_JOBID
              FROM HR.JOBS
                   INNER JOIN HR.EMPLOYEES ON (JOBS.JOB_ID =
EMPLOYEES.JOB_ID)
             WHERE JOB_TITLE LIKE '%Marketing%'
          GROUP BY EMPLOYEES.JOB_ID) Q3
         CROSS JOIN (  SELECT T1.JOB_ID, COUNT (*) MAS_AVG
                         FROM (SELECT JOB_ID, SALARY
                                 FROM HR.EMPLOYEES
                                WHERE JOB_ID LIKE '%MK%') T1
                              INNER JOIN
                              (  SELECT DISTINCT JOB_ID, AVG
(SALARY) AVG_SAL_JOBID
                                   FROM HR.EMPLOYEES
                                  WHERE JOB_ID LIKE '%MK%'
```

```
                                    GROUP BY JOB_ID) T2
                                      ON      (SALARY > AVG_SAL_JOBID)
                                         AND (T1.JOB_ID = T2.JOB_ID)
                          GROUP BY T1.JOB_ID) Q6
          CROSS JOIN
          (  SELECT DISTINCT EMPLOYEES.JOB_ID, MIN (SALARY)
MIN_SAL_JOBID
               FROM HR.JOBS
                    INNER JOIN HR.EMPLOYEES ON (JOBS.JOB_ID =
EMPLOYEES.JOB_ID)
              WHERE JOB_TITLE LIKE '%Marketing%'
           GROUP BY EMPLOYEES.JOB_ID) Q4
          CROSS JOIN
          (  SELECT R1.JOB_ID, COUNT (*) MENOS_AVG
               FROM (SELECT JOB_ID, SALARY
                       FROM HR.EMPLOYEES
                      WHERE JOB_ID LIKE '%MK%') R1
                    INNER JOIN
                    (  SELECT DISTINCT JOB_ID, AVG (SALARY)
AVG_SAL_JOBID
                         FROM HR.EMPLOYEES
                        WHERE JOB_ID LIKE '%MK%'
                     GROUP BY JOB_ID) R2
                       ON (R1.JOB_ID = R2.JOB_ID)
             WHERE (SALARY < AVG_SAL_JOBID) OR (SALARY =
AVG_SAL_JOBID)
           GROUP BY R1.JOB_ID) Q7
          CROSS JOIN (  SELECT DISTINCT JOB_ID, COUNT
(EMPLOYEE_ID) TOTAL_JOBID
                          FROM HR.EMPLOYEES
                         WHERE JOB_ID LIKE '%MK%'
                      GROUP BY JOB_ID) Q5
ORDER BY DEPARTMENT_ID, Q2.JOB_ID ASC;
```

- Mejora c2_p3a_toad: Alternativa 186 (mejor tiempo)

```
select q1.region_name,
       q1.department_id,
       q2.job_id,
       q2.AVG_Sal_JobId,
       q3.MAX_Sal_JobId,
       q6.Mas_AVG,
       q4.MIN_Sal_JobId,
       q7.Menos_AVG,
       q5.Total_JobId
  from (select /*+ FULL(DEPARTMENTS) */ distinct regions.region_name,
               departments.department_id,
               JOBS1.job_id
          FROM hr.locations,
               hr.departments,
               hr.employees EMPLOYEES1,
               hr.regions,
               hr.countries,
               hr.jobs JOBS1
         where JOBS1.job_title like '%Marketing%'
           and JOBS1.job_id = EMPLOYEES1.job_id || ''
           and EMPLOYEES1.department_id = departments.department_id +
0
```

```
           and departments.location_id = locations.location_id + 0
           and countries.country_id = locations.country_id || ''
           and countries.region_id = regions.region_id) q1,
       (select distinct EMPLOYEES2.job_id,
               avg(salary) as AVG_Sal_JobId
          from hr.employees EMPLOYEES2,
               hr.jobs JOBS2
         where JOBS2.job_title like '%Marketing%'
           and JOBS2.job_id = EMPLOYEES2.job_id
         group by EMPLOYEES2.job_id) q2,
       (select distinct EMPLOYEES3.job_id,
               max(salary) as MAX_Sal_JobId
          from hr.employees EMPLOYEES3,
               hr.jobs JOBS3
         where JOBS3.job_title like '%Marketing%'
           and JOBS3.job_id = EMPLOYEES3.job_id
         group by EMPLOYEES3.job_id) q3,
       (select t1.job_id,
               count(*) as Mas_AVG
          from (select job_id,
                       salary
                  from hr.employees EMPLOYEES4
                 where job_id like '%MK%') t1,
               (select distinct job_id,
                       avg(salary) as AVG_Sal_JobId
                  from hr.employees EMPLOYEES5
                 where job_id like '%MK%'
                 group by job_id) t2
         where t1.job_id = t2.job_id
           and t1.salary > t2.AVG_Sal_JobId
         group by t1.job_id) q6,
       (select distinct EMPLOYEES6.job_id,
               min(salary) as MIN_Sal_JobId
          from hr.employees EMPLOYEES6,
               hr.jobs JOBS4
         where JOBS4.job_title like '%Marketing%'
           and JOBS4.job_id = EMPLOYEES6.job_id
         group by EMPLOYEES6.job_id) q4,
       (select r1.job_id,
               count(*) as Menos_AVG
          from (select job_id,
                       salary
                  from hr.employees EMPLOYEES7
                 where job_id like '%MK%') r1,
               (select distinct job_id,
                       avg(salary) as AVG_Sal_JobId
                  from hr.employees EMPLOYEES8
                 where job_id like '%MK%'
                 group by job_id) r2
         where r1.job_id = r2.job_id
           and (r1.salary < r2.AVG_Sal_JobId
                or r1.salary = r2.AVG_Sal_JobId)
         group by r1.job_id) q7,
       (select distinct job_id,
               count(employee_id) as Total_JobId
          from hr.employees EMPLOYEES9
         where job_id like '%MK%'
         group by job_id) q5
 where q1.job_id = q2.job_id
   and q1.job_id = q3.job_id
   and q1.job_id = q4.job_id
```

```
   and q1.job_id = q5.job_id
   and q1.job_id = q6.job_id
   and q1.job_id = q7.job_id
 order by q1.department_id, q2.job_id asc;
```

- Mejora c2_p3b_toad: Alternativa 120 (menor coste)

```
select q1.region_name,
       q1.department_id,
       q2.job_id,
       q2.AVG_Sal_JobId,
       q3.MAX_Sal_JobId,
       q6.Mas_AVG,
       q4.MIN_Sal_JobId,
       q7.Menos_AVG,
       q5.Total_JobId
  from (select distinct regions.region_name,
                departments.department_id,
                JOBS1.job_id
           FROM hr.employees EMPLOYEES1,
                hr.regions,
                hr.countries,
                hr.locations,
                hr.departments,
                hr.jobs JOBS1
          where JOBS1.job_title like '%Marketing%'
            and JOBS1.job_id = EMPLOYEES1.job_id || ''
            and departments.department_id = EMPLOYEES1.department_id +
0
            and departments.location_id = locations.location_id + 0
            and locations.country_id = countries.country_id || ''
            and countries.region_id = regions.region_id) q1,
        (select distinct EMPLOYEES2.job_id,
                avg(salary) as AVG_Sal_JobId
           from hr.employees EMPLOYEES2,
                hr.jobs JOBS2
          where JOBS2.job_title like '%Marketing%'
            and JOBS2.job_id = EMPLOYEES2.job_id
          group by EMPLOYEES2.job_id) q2,
        (select distinct EMPLOYEES3.job_id,
                max(salary) as MAX_Sal_JobId
           from hr.employees EMPLOYEES3,
                hr.jobs JOBS3
          where JOBS3.job_title like '%Marketing%'
            and JOBS3.job_id = EMPLOYEES3.job_id
          group by EMPLOYEES3.job_id) q3,
        (select /*+ NO_USE_MERGE(T2,T1) */ t1.job_id,
                count(*) as Mas_AVG
           from (select job_id,
                        salary
                   from hr.employees EMPLOYEES4
                  where job_id like '%MK%') t1,
                (select distinct job_id,
                        avg(salary) as AVG_Sal_JobId
                   from hr.employees EMPLOYEES5
                  where job_id like '%MK%'
                  group by job_id) t2
          where t1.job_id = t2.job_id
            and t1.salary > t2.AVG_Sal_JobId
          group by t1.job_id) q6,
        (select distinct EMPLOYEES6.job_id,
```

```
               min(salary) as MIN_Sal_JobId
          from hr.employees EMPLOYEES6,
               hr.jobs JOBS4
         where JOBS4.job_title like '%Marketing%'
           and JOBS4.job_id = EMPLOYEES6.job_id
         group by EMPLOYEES6.job_id) q4,
       (select r1.job_id,
               count(*) as Menos_AVG
          from (select job_id,
                       salary
                  from hr.employees EMPLOYEES7
                 where job_id like '%MK%') r1,
               (select distinct job_id,
                       avg(salary) as AVG_Sal_JobId
                  from hr.employees EMPLOYEES8
                 where job_id like '%MK%'
                 group by job_id) r2
         where r1.job_id = r2.job_id
           and (r1.salary < r2.AVG_Sal_JobId
                 or r1.salary = r2.AVG_Sal_JobId)
         group by r1.job_id) q7,
       (select distinct job_id,
               count(employee_id) as Total_JobId
          from hr.employees EMPLOYEES9
         where job_id like '%MK%'
         group by job_id) q5
 where q1.job_id = q2.job_id
   and q1.job_id = q3.job_id
   and q1.job_id = q4.job_id
   and q1.job_id = q5.job_id
   and q1.job_id = q6.job_id
   and q1.job_id = q7.job_id
 order by q1.department_id, q2.job_id asc;
```

- Mejora c2_p5a_toad: Alternativa 6 (mejor tiempo); Adv SQL Opt

```
select q1.region_name,
       q1.department_id,
       q2.job_id,
       q2.AVG_Sal_JobId,
       q3.MAX_Sal_JobId,
       q6.Mas_AVG,
       q4.MIN_Sal_JobId,
       q7.Menos_AVG,
       q5.Total_JobId
  from (select /*+ FULL(DEPARTMENTS) */ distinct regions.region_name,
               departments.department_id,
               JOBS1.job_id
          from hr.jobs JOBS1,
               hr.employees EMPLOYEES1,
               hr.departments,
               hr.locations,
               hr.countries,
               hr.regions
         where JOBS1.job_title like '%Marketing%'
           and JOBS1.job_id = EMPLOYEES1.job_id
           and EMPLOYEES1.department_id = departments.department_id
           and departments.location_id = locations.location_id
           and locations.country_id = countries.country_id
           and countries.region_id = regions.region_id) q1,
       (select distinct EMPLOYEES2.job_id,
```

```
               avg(salary) as AVG_Sal_JobId
          from hr.employees EMPLOYEES2,
               hr.jobs JOBS2
         where JOBS2.job_title like '%Marketing%'
           and JOBS2.job_id = EMPLOYEES2.job_id
         group by EMPLOYEES2.job_id) q2,
       (select distinct EMPLOYEES3.job_id,
               max(salary) as MAX_Sal_JobId
          from hr.employees EMPLOYEES3,
               hr.jobs JOBS3
         where JOBS3.job_title like '%Marketing%'
           and JOBS3.job_id = EMPLOYEES3.job_id
         group by EMPLOYEES3.job_id) q3,
       (select t1.job_id,
               count(*) as Mas_AVG
          from (select job_id,
                       salary
                  from hr.employees EMPLOYEES4
                 where job_id like '%MK%') t1,
               (select distinct job_id,
                       avg(salary) as AVG_Sal_JobId
                  from hr.employees EMPLOYEES5
                 where job_id like '%MK%'
                 group by job_id) t2
         where t1.job_id = t2.job_id
           and t1.salary > t2.AVG_Sal_JobId
         group by t1.job_id) q6,
       (select distinct EMPLOYEES6.job_id,
               min(salary) as MIN_Sal_JobId
          from hr.employees EMPLOYEES6,
               hr.jobs JOBS4
         where JOBS4.job_title like '%Marketing%'
           and JOBS4.job_id = EMPLOYEES6.job_id
         group by EMPLOYEES6.job_id) q4,
       (select r1.job_id,
               count(*) as Menos_AVG
          from (select job_id,
                       salary
                  from hr.employees EMPLOYEES7
                 where job_id like '%MK%') r1,
               (select distinct job_id,
                       avg(salary) as AVG_Sal_JobId
                  from hr.employees EMPLOYEES8
                 where job_id like '%MK%'
                 group by job_id) r2
         where r1.job_id = r2.job_id
           and (r1.salary < r2.AVG_Sal_JobId
             or r1.salary = r2.AVG_Sal_JobId)
         group by r1.job_id) q7,
       (select distinct job_id,
               count(employee_id) as Total_JobId
          from hr.employees EMPLOYEES9
         where job_id like '%MK%'
         group by job_id) q5
where q1.job_id = q2.job_id
  and q1.job_id = q3.job_id
  and q1.job_id = q4.job_id
  and q1.job_id = q5.job_id
  and q1.job_id = q6.job_id
  and q1.job_id = q7.job_id
order by q1.department_id, q2.job_id asc;
```

- Mejora c2_p5b_toad: Alternativa 16 (menor coste); Adv SQL Opt

```
select q1.region_name,
       q1.department_id,
       q2.job_id,
       q2.AVG_Sal_JobId,
       q3.MAX_Sal_JobId,
       q6.Mas_AVG,
       q4.MIN_Sal_JobId,
       q7.Menos_AVG,
       q5.Total_JobId
  from (select /*+ INDEX(REGIONS) */ distinct regions.region_name,
               departments.department_id,
               JOBS1.job_id
          from hr.jobs JOBS1,
               hr.employees EMPLOYEES1,
               hr.departments,
               hr.locations,
               hr.countries,
               hr.regions
         where JOBS1.job_title like '%Marketing%'
           and JOBS1.job_id = EMPLOYEES1.job_id
           and EMPLOYEES1.department_id = departments.department_id
           and departments.location_id = locations.location_id
           and locations.country_id = countries.country_id
           and countries.region_id = regions.region_id) q1,
       (select distinct EMPLOYEES2.job_id,
               avg(salary) as AVG_Sal_JobId
          from hr.employees EMPLOYEES2,
               hr.jobs JOBS2
         where JOBS2.job_title like '%Marketing%'
           and JOBS2.job_id = EMPLOYEES2.job_id
         group by EMPLOYEES2.job_id) q2,
       (select distinct EMPLOYEES3.job_id,
               max(salary) as MAX_Sal_JobId
          from hr.employees EMPLOYEES3,
               hr.jobs JOBS3
         where JOBS3.job_title like '%Marketing%'
           and JOBS3.job_id = EMPLOYEES3.job_id
         group by EMPLOYEES3.job_id) q3,
       (select t1.job_id,
               count(*) as Mas_AVG
          from (select job_id,
                       salary
                  from hr.employees EMPLOYEES4
                 where job_id like '%MK%') t1,
               (select distinct job_id,
                       avg(salary) as AVG_Sal_JobId
                  from hr.employees EMPLOYEES5
                 where job_id like '%MK%'
                 group by job_id) t2
         where t1.job_id = t2.job_id
           and t1.salary > t2.AVG_Sal_JobId
         group by t1.job_id) q6,
       (select distinct EMPLOYEES6.job_id,
               min(salary) as MIN_Sal_JobId
          from hr.employees EMPLOYEES6,
               hr.jobs JOBS4
         where JOBS4.job_title like '%Marketing%'
           and JOBS4.job_id = EMPLOYEES6.job_id
         group by EMPLOYEES6.job_id) q4,
```

```
       (select r1.job_id,
               count(*) as Menos_AVG
          from (select job_id,
                       salary
                  from hr.employees EMPLOYEES7
                 where job_id like '%MK%') r1,
               (select distinct job_id,
                       avg(salary) as AVG_Sal_JobId
                  from hr.employees EMPLOYEES8
                 where job_id like '%MK%'
                 group by job_id) r2
         where r1.job_id = r2.job_id
           and (r1.salary < r2.AVG_Sal_JobId
                or r1.salary = r2.AVG_Sal_JobId)
         group by r1.job_id) q7,
       (select distinct job_id,
               count(employee_id) as Total_JobId
          from hr.employees EMPLOYEES9
         where job_id like '%MK%'
         group by job_id) q5
 where q1.job_id = q2.job_id
   and q1.job_id = q3.job_id
   and q1.job_id = q4.job_id
   and q1.job_id = q5.job_id
   and q1.job_id = q6.job_id
   and q1.job_id = q7.job_id
 order by q1.department_id, q2.job_id asc;
```

- Mejora c2_p6a_toad: Alternativa 10 (mejor tiempo); Auto Opt SQL

```
SELECT /*+ ORDERED */ q1.region_name,
       q1.department_id,
       q2.job_id,
       q2.AVG_Sal_JobId,
       q3.MAX_Sal_JobId,
       q6.Mas_AVG,
       q4.MIN_Sal_JobId,
       q7.Menos_AVG,
       q5.Total_JobId
  FROM (SELECT DISTINCT regions.region_name,
               departments.department_id,
               JOBS1.job_id
          FROM hr.jobs JOBS1,
               hr.employees EMPLOYEES1,
               hr.departments,
               hr.locations,
               hr.countries,
               hr.regions
         WHERE JOBS1.job_title LIKE '%Marketing%'
           AND JOBS1.job_id = EMPLOYEES1.job_id
           AND EMPLOYEES1.department_id =
departments.department_id
           AND departments.location_id = locations.location_id
           AND locations.country_id = countries.country_id
           AND countries.region_id = regions.region_id) q1,
       (SELECT DISTINCT EMPLOYEES2.job_id,
               AVG(salary) AS AVG_Sal_JobId
          FROM hr.employees EMPLOYEES2,
```

```
            hr.jobs JOBS2
      WHERE JOBS2.job_title LIKE '%Marketing%'
        AND JOBS2.job_id = EMPLOYEES2.job_id
      GROUP BY EMPLOYEES2.job_id) q2,
     (SELECT DISTINCT EMPLOYEES3.job_id,
            MAX(salary) AS MAX_Sal_JobId
       FROM hr.employees EMPLOYEES3,
            hr.jobs JOBS3
      WHERE JOBS3.job_title LIKE '%Marketing%'
        AND JOBS3.job_id = EMPLOYEES3.job_id
      GROUP BY EMPLOYEES3.job_id) q3,
     (SELECT t1.job_id,
            COUNT(*) AS Mas_AVG
       FROM (SELECT job_id,
                    salary
               FROM hr.employees EMPLOYEES4
              WHERE job_id LIKE '%MK%') t1,
            (SELECT DISTINCT job_id,
                    AVG(salary) AS AVG_Sal_JobId
               FROM hr.employees EMPLOYEES5
              WHERE job_id LIKE '%MK%'
              GROUP BY job_id) t2
      WHERE t1.job_id = t2.job_id
        AND t1.salary > t2.AVG_Sal_JobId
      GROUP BY t1.job_id) q6,
     (SELECT DISTINCT EMPLOYEES6.job_id,
            MIN(salary) AS MIN_Sal_JobId
       FROM hr.employees EMPLOYEES6,
            hr.jobs JOBS4
      WHERE JOBS4.job_title LIKE '%Marketing%'
        AND JOBS4.job_id = EMPLOYEES6.job_id
      GROUP BY EMPLOYEES6.job_id) q4,
     (SELECT r1.job_id,
            COUNT(*) AS Menos_AVG
       FROM (SELECT job_id,
                    salary
               FROM hr.employees EMPLOYEES7
              WHERE job_id LIKE '%MK%') r1,
            (SELECT DISTINCT job_id,
                    AVG(salary) AS AVG_Sal_JobId
               FROM hr.employees EMPLOYEES8
              WHERE job_id LIKE '%MK%'
              GROUP BY job_id) r2
      WHERE r1.job_id = r2.job_id
        AND (r1.salary < r2.AVG_Sal_JobId
             OR r1.salary = r2.AVG_Sal_JobId)
      GROUP BY r1.job_id) q7,
     (SELECT DISTINCT job_id,
            COUNT(employee_id) AS Total_JobId
       FROM hr.employees EMPLOYEES9
      WHERE job_id LIKE '%MK%'
      GROUP BY job_id) q5
WHERE q1.job_id = q2.job_id
  AND q1.job_id = q3.job_id
  AND q1.job_id = q4.job_id
  AND q1.job_id = q5.job_id
```

```
   AND q1.job_id = q6.job_id
   AND q1.job_id = q7.job_id
 ORDER BY q1.department_id, q2.job_id ASC;
```

- Mejora c2_p6b_toad: Alternativa 190 (menor coste)

```
SELECT q1.region_name,
       q1.department_id,
       q2.job_id,
       q2.AVG_Sal_JobId,
       q3.MAX_Sal_JobId,
       q6.Mas_AVG,
       q4.MIN_Sal_JobId,
       q7.Menos_AVG,
       q5.Total_JobId
  FROM (SELECT DISTINCT regions.region_name,
               departments.department_id,
               JOBS1.job_id
          FROM hr.jobs JOBS1,
               hr.employees EMPLOYEES1,
               hr.departments,
               hr.locations,
               hr.countries,
               hr.regions
         WHERE JOBS1.job_title LIKE '%Marketing%'
           AND EMPLOYEES1.job_id = JOBS1.job_id
           AND departments.department_id =
EMPLOYEES1.department_id
           AND locations.location_id = departments.location_id
           AND countries.country_id = locations.country_id
           AND regions.region_id = countries.region_id) q1,
       (SELECT DISTINCT EMPLOYEES2.job_id,
               AVG(salary) AS AVG_Sal_JobId
          FROM hr.employees EMPLOYEES2,
               hr.jobs JOBS2
         WHERE JOBS2.job_title LIKE '%Marketing%'
           AND EMPLOYEES2.job_id = JOBS2.job_id
         GROUP BY EMPLOYEES2.job_id) q2,
       (SELECT DISTINCT EMPLOYEES3.job_id,
               MAX(salary) AS MAX_Sal_JobId
          FROM hr.employees EMPLOYEES3,
               hr.jobs JOBS3
         WHERE JOBS3.job_title LIKE '%Marketing%'
           AND EMPLOYEES3.job_id = JOBS3.job_id
         GROUP BY EMPLOYEES3.job_id) q3,
       (SELECT t1.job_id,
               COUNT(*) AS Mas_AVG
          FROM (SELECT job_id,
                       salary
                  FROM hr.employees EMPLOYEES4
                 WHERE job_id LIKE '%MK%') t1,
               (SELECT DISTINCT job_id,
                       AVG(salary) AS AVG_Sal_JobId
                  FROM hr.employees EMPLOYEES5
                 WHERE job_id LIKE '%MK%'
                 GROUP BY job_id) t2
```

```
          WHERE t2.job_id = t1.job_id
            AND t2.AVG_Sal_JobId < t1.salary
          GROUP BY t1.job_id) q6,
        (SELECT DISTINCT EMPLOYEES6.job_id,
                MIN(salary) AS MIN_Sal_JobId
           FROM hr.employees EMPLOYEES6,
                hr.jobs JOBS4
          WHERE JOBS4.job_title LIKE '%Marketing%'
            AND EMPLOYEES6.job_id = JOBS4.job_id
          GROUP BY EMPLOYEES6.job_id) q4,
        (SELECT r1.job_id,
                COUNT(*) AS Menos_AVG
           FROM (SELECT job_id,
                        salary
                   FROM hr.employees EMPLOYEES7
                  WHERE job_id LIKE '%MK%') r1,
                (SELECT DISTINCT job_id,
                        AVG(salary) AS AVG_Sal_JobId
                   FROM hr.employees EMPLOYEES8
                  WHERE job_id LIKE '%MK%'
                  GROUP BY job_id) r2
          WHERE r2.job_id = r1.job_id
            AND (r2.AVG_Sal_JobId > r1.salary
                  OR r2.AVG_Sal_JobId = r1.salary)
          GROUP BY r1.job_id) q7,
        (SELECT DISTINCT job_id,
                COUNT(employee_id) AS Total_JobId
           FROM hr.employees EMPLOYEES9
          WHERE job_id LIKE '%MK%'
          GROUP BY job_id) q5
 WHERE q2.job_id = q1.job_id
   AND q3.job_id = q1.job_id
   AND q4.job_id = q1.job_id
   AND q5.job_id = q1.job_id
   AND q6.job_id = q1.job_id
   AND q7.job_id = q1.job_id
   AND q3.job_id = q2.job_id
   AND q4.job_id = q2.job_id
   AND q5.job_id = q2.job_id
   AND q6.job_id = q2.job_id
   AND q7.job_id = q2.job_id
   AND q4.job_id = q3.job_id
   AND q5.job_id = q3.job_id
   AND q6.job_id = q3.job_id
   AND q7.job_id = q3.job_id
   AND q5.job_id = q4.job_id
   AND q6.job_id = q4.job_id
   AND q7.job_id = q4.job_id
   AND q6.job_id = q5.job_id
   AND q7.job_id = q5.job_id
   AND q7.job_id = q6.job_id
 ORDER BY q1.department_id, q2.job_id ASC;
```

- Mejora c2_p7a_toad: Alternativa 188 (mejor tiempo)

```
select q1.region_name,
```

```
       q1.department_id,
       q2.job_id,
       q2.AVG_Sal_JobId,
       q3.MAX_Sal_JobId,
       q6.Mas_AVG,
       q4.MIN_Sal_JobId,
       q7.Menos_AVG,
       q5.Total_JobId
  from (select /*+ INDEX_JOIN(REGIONS) */ distinct
regions.region_name,
               departments.department_id,
               JOBS1.job_id
          FROM hr.locations,
               hr.departments,
               hr.employees EMPLOYEES1,
               hr.regions,
               hr.countries,
               hr.jobs JOBS1
         where JOBS1.job_title like '%Marketing%'
           and JOBS1.job_id = EMPLOYEES1.job_id || ''
           and EMPLOYEES1.department_id = departments.department_id +
0
           and departments.location_id = locations.location_id + 0
           and countries.country_id = locations.country_id || ''
           and countries.region_id = regions.region_id) q1,
       (select distinct EMPLOYEES2.job_id,
               avg(salary) as AVG_Sal_JobId
          from hr.employees EMPLOYEES2,
               hr.jobs JOBS2
         where JOBS2.job_title like '%Marketing%'
           and JOBS2.job_id = EMPLOYEES2.job_id
         group by EMPLOYEES2.job_id) q2,
       (select distinct EMPLOYEES3.job_id,
               max(salary) as MAX_Sal_JobId
          from hr.employees EMPLOYEES3,
               hr.jobs JOBS3
         where JOBS3.job_title like '%Marketing%'
           and JOBS3.job_id = EMPLOYEES3.job_id
         group by EMPLOYEES3.job_id) q3,
       (select t1.job_id,
               count(*) as Mas_AVG
          from (select job_id,
                       salary
                  from hr.employees EMPLOYEES4
                 where job_id like '%MK%') t1,
               (select distinct job_id,
                       avg(salary) as AVG_Sal_JobId
                  from hr.employees EMPLOYEES5
                 where job_id like '%MK%'
                 group by job_id) t2
         where t1.job_id = t2.job_id
           and t1.salary > t2.AVG_Sal_JobId
         group by t1.job_id) q6,
       (select distinct EMPLOYEES6.job_id,
               min(salary) as MIN_Sal_JobId
          from hr.employees EMPLOYEES6,
               hr.jobs JOBS4
         where JOBS4.job_title like '%Marketing%'
           and JOBS4.job_id = EMPLOYEES6.job_id
         group by EMPLOYEES6.job_id) q4,
       (select r1.job_id,
```

```
                count(*) as Menos_AVG
           from (select job_id,
                        salary
                   from hr.employees EMPLOYEES7
                  where job_id like '%MK%') r1,
                (select distinct job_id,
                        avg(salary) as AVG_Sal_JobId
                   from hr.employees EMPLOYEES8
                  where job_id like '%MK%'
                  group by job_id) r2
          where r1.job_id = r2.job_id
            and (r1.salary < r2.AVG_Sal_JobId
                 or r1.salary = r2.AVG_Sal_JobId)
          group by r1.job_id) q7,
        (select distinct job_id,
                count(employee_id) as Total_JobId
           from hr.employees EMPLOYEES9
          where job_id like '%MK%'
          group by job_id) q5
 where q1.job_id = q2.job_id
   and q1.job_id = q3.job_id
   and q1.job_id = q4.job_id
   and q1.job_id = q5.job_id
   and q1.job_id = q6.job_id
   and q1.job_id = q7.job_id
 order by q1.department_id, q2.job_id asc;
```

- Mejora c2_p7b_toad: Alternativa 8 (menor coste)

```
select q1.region_name,
       q1.department_id,
       q2.job_id,
       q2.AVG_Sal_JobId,
       q3.MAX_Sal_JobId,
       q6.Mas_AVG,
       q4.MIN_Sal_JobId,
       q7.Menos_AVG,
       q5.Total_JobId
  from (select /*+ INDEX_JOIN(REGIONS) */ distinct
regions.region_name,
               departments.department_id,
               JOBS1.job_id
          from hr.jobs JOBS1,
               hr.employees EMPLOYEES1,
               hr.departments,
               hr.locations,
               hr.countries,
               hr.regions
         where JOBS1.job_title like '%Marketing%'
           and JOBS1.job_id = EMPLOYEES1.job_id
           and EMPLOYEES1.department_id = departments.department_id
           and departments.location_id = locations.location_id
           and locations.country_id = countries.country_id
           and countries.region_id = regions.region_id) q1,
       (select distinct EMPLOYEES2.job_id,
               avg(salary) as AVG_Sal_JobId
          from hr.employees EMPLOYEES2,
               hr.jobs JOBS2
         where JOBS2.job_title like '%Marketing%'
```

```
           and JOBS2.job_id = EMPLOYEES2.job_id
         group by EMPLOYEES2.job_id) q2,
       (select distinct EMPLOYEES3.job_id,
               max(salary) as MAX_Sal_JobId
          from hr.employees EMPLOYEES3,
               hr.jobs JOBS3
         where JOBS3.job_title like '%Marketing%'
           and JOBS3.job_id = EMPLOYEES3.job_id
         group by EMPLOYEES3.job_id) q3,
       (select t1.job_id,
               count(*) as Mas_AVG
          from (select job_id,
                       salary
                  from hr.employees EMPLOYEES4
                 where job_id like '%MK%') t1,
               (select distinct job_id,
                       avg(salary) as AVG_Sal_JobId
                  from hr.employees EMPLOYEES5
                 where job_id like '%MK%'
                 group by job_id) t2
         where t1.job_id = t2.job_id
           and t1.salary > t2.AVG_Sal_JobId
         group by t1.job_id) q6,
       (select distinct EMPLOYEES6.job_id,
               min(salary) as MIN_Sal_JobId
          from hr.employees EMPLOYEES6,
               hr.jobs JOBS4
         where JOBS4.job_title like '%Marketing%'
           and JOBS4.job_id = EMPLOYEES6.job_id
         group by EMPLOYEES6.job_id) q4,
       (select r1.job_id,
               count(*) as Menos_AVG
          from (select job_id,
                       salary
                  from hr.employees EMPLOYEES7
                 where job_id like '%MK%') r1,
               (select distinct job_id,
                       avg(salary) as AVG_Sal_JobId
                  from hr.employees EMPLOYEES8
                 where job_id like '%MK%'
                 group by job_id) r2
         where r1.job_id = r2.job_id
           and (r1.salary < r2.AVG_Sal_JobId
                 or r1.salary = r2.AVG_Sal_JobId)
         group by r1.job_id) q7,
       (select distinct job_id,
               count(employee_id) as Total_JobId
          from hr.employees EMPLOYEES9
         where job_id like '%MK%'
         group by job_id) q5
where q1.job_id = q2.job_id
  and q1.job_id = q3.job_id
  and q1.job_id = q4.job_id
  and q1.job_id = q5.job_id
  and q1.job_id = q6.job_id
  and q1.job_id = q7.job_id
order by q1.department_id, q2.job_id asc;
```

- Mejora c2_p8a_toad: Alternativa 32 (mejor tiempo)

```
select q1.region_name,
       q1.department_id,
       q2.job_id,
       q2.AVG_Sal_JobId,
       q3.MAX_Sal_JobId,
       q6.Mas_AVG,
       q4.MIN_Sal_JobId,
       q7.Menos_AVG,
       q5.Total_JobId
  from (select /*+ INDEX_DESC(LOCATIONS) */ distinct
regions.region_name,
               departments.department_id,
               JOBS1.job_id
          from hr.jobs JOBS1,
               hr.employees EMPLOYEES1,
               hr.departments,
               hr.locations,
               hr.countries,
               hr.regions
         where JOBS1.job_title like '%Marketing%'
           and JOBS1.job_id = EMPLOYEES1.job_id
           and EMPLOYEES1.department_id = departments.department_id
           and departments.location_id = locations.location_id
           and locations.country_id = countries.country_id
           and countries.region_id = regions.region_id) q1,
       (select distinct EMPLOYEES2.job_id,
               avg(salary) as AVG_Sal_JobId
          from hr.employees EMPLOYEES2,
               hr.jobs JOBS2
         where JOBS2.job_title like '%Marketing%'
           and JOBS2.job_id = EMPLOYEES2.job_id
         group by EMPLOYEES2.job_id) q2,
       (select distinct EMPLOYEES3.job_id,
               max(salary) as MAX_Sal_JobId
          from hr.employees EMPLOYEES3,
               hr.jobs JOBS3
         where JOBS3.job_title like '%Marketing%'
           and JOBS3.job_id = EMPLOYEES3.job_id
         group by EMPLOYEES3.job_id) q3,
       (select t1.job_id,
               count(*) as Mas_AVG
          from (select job_id,
                       salary
                  from hr.employees EMPLOYEES4
                 where job_id like '%MK%') t1,
               (select distinct job_id,
                       avg(salary) as AVG_Sal_JobId
                  from hr.employees EMPLOYEES5
                 where job_id like '%MK%'
                 group by job_id) t2
         where t1.job_id = t2.job_id
           and t1.salary > t2.AVG_Sal_JobId
         group by t1.job_id) q6,
       (select distinct EMPLOYEES6.job_id,
               min(salary) as MIN_Sal_JobId
          from hr.employees EMPLOYEES6,
               hr.jobs JOBS4
         where JOBS4.job_title like '%Marketing%'
           and JOBS4.job_id = EMPLOYEES6.job_id
```

```
          group by EMPLOYEES6.job_id) q4,
        (select r1.job_id,
                count(*) as Menos_AVG
           from (select job_id,
                        salary
                   from hr.employees EMPLOYEES7
                  where job_id like '%MK%') r1,
                (select distinct job_id,
                        avg(salary) as AVG_Sal_JobId
                   from hr.employees EMPLOYEES8
                  where job_id like '%MK%'
                  group by job_id) r2
          where r1.job_id = r2.job_id
            and (r1.salary < r2.AVG_Sal_JobId
                 or r1.salary = r2.AVG_Sal_JobId)
          group by r1.job_id) q7,
        (select distinct job_id,
                count(employee_id) as Total_JobId
           from hr.employees EMPLOYEES9
          where job_id like '%MK%'
          group by job_id) q5
 where q1.job_id = q2.job_id
   and q1.job_id = q3.job_id
   and q1.job_id = q4.job_id
   and q1.job_id = q5.job_id
   and q1.job_id = q6.job_id
   and q1.job_id = q7.job_id
 order by q1.department_id, q2.job_id asc;
```

- Mejora c2_p8b_toad: Alternativa 35 (menor coste)

```
select q1.region_name,
       q1.department_id,
       q2.job_id,
       q2.AVG_Sal_JobId,
       q3.MAX_Sal_JobId,
       q6.Mas_AVG,
       q4.MIN_Sal_JobId,
       q7.Menos_AVG,
       q5.Total_JobId
  from (select /*+ INDEX_DESC(REGIONS) */ distinct
regions.region_name,
               departments.department_id,
               JOBS1.job_id
          from hr.jobs JOBS1,
               hr.employees EMPLOYEES1,
               hr.departments,
               hr.locations,
               hr.countries,
               hr.regions
         where JOBS1.job_title like '%Marketing%'
           and JOBS1.job_id = EMPLOYEES1.job_id
           and EMPLOYEES1.department_id = departments.department_id
           and departments.location_id = locations.location_id
           and locations.country_id = countries.country_id
           and countries.region_id = regions.region_id) q1,
       (select distinct EMPLOYEES2.job_id,
               avg(salary) as AVG_Sal_JobId
          from hr.employees EMPLOYEES2,
               hr.jobs JOBS2
         where JOBS2.job_title like '%Marketing%'
```

```
           and JOBS2.job_id = EMPLOYEES2.job_id
         group by EMPLOYEES2.job_id) q2,
       (select distinct EMPLOYEES3.job_id,
               max(salary) as MAX_Sal_JobId
          from hr.employees EMPLOYEES3,
               hr.jobs JOBS3
         where JOBS3.job_title like '%Marketing%'
           and JOBS3.job_id = EMPLOYEES3.job_id
         group by EMPLOYEES3.job_id) q3,
       (select t1.job_id,
               count(*) as Mas_AVG
          from (select job_id,
                       salary
                  from hr.employees EMPLOYEES4
                 where job_id like '%MK%') t1,
               (select distinct job_id,
                       avg(salary) as AVG_Sal_JobId
                  from hr.employees EMPLOYEES5
                 where job_id like '%MK%'
                 group by job_id) t2
         where t1.job_id = t2.job_id
           and t1.salary > t2.AVG_Sal_JobId
         group by t1.job_id) q6,
       (select distinct EMPLOYEES6.job_id,
               min(salary) as MIN_Sal_JobId
          from hr.employees EMPLOYEES6,
               hr.jobs JOBS4
         where JOBS4.job_title like '%Marketing%'
           and JOBS4.job_id = EMPLOYEES6.job_id
         group by EMPLOYEES6.job_id) q4,
       (select r1.job_id,
               count(*) as Menos_AVG
          from (select job_id,
                       salary
                  from hr.employees EMPLOYEES7
                 where job_id like '%MK%') r1,
               (select distinct job_id,
                       avg(salary) as AVG_Sal_JobId
                  from hr.employees EMPLOYEES8
                 where job_id like '%MK%'
                 group by job_id) r2
         where r1.job_id = r2.job_id
           and (r1.salary < r2.AVG_Sal_JobId
                 or r1.salary = r2.AVG_Sal_JobId)
         group by r1.job_id) q7,
       (select distinct job_id,
               count(employee_id) as Total_JobId
          from hr.employees EMPLOYEES9
         where job_id like '%MK%'
         group by job_id) q5
where q1.job_id = q2.job_id
  and q1.job_id = q3.job_id
  and q1.job_id = q4.job_id
  and q1.job_id = q5.job_id
  and q1.job_id = q6.job_id
  and q1.job_id = q7.job_id
order by q1.department_id, q2.job_id asc;
```

Consulta 3:

- Mejora c3_p1_toad:

```
SELECT JOB_ID,
       DEPARTMENT_ID,
       COUNTRY_NAME,
       EMPL_CONTRATADOS,
       TOTAL_JOBID
  FROM (SELECT Q1.JOB_ID,DEPARTMENT_ID,COUNTRY_NAME,
EMPL_CONTRATADOS, TOTAL_JOBID
            FROM (SELECT DISTINCT JOB_ID,
EMPLOYEES.DEPARTMENT_ID, COUNTRY_NAME
                    FROM HR.EMPLOYEES,HR.DEPARTMENTS,
HR.LOCATIONS,HR.COUNTRIES
                   WHERE     EMPLOYEES.DEPARTMENT_ID =
                                DEPARTMENTS.DEPARTMENT_ID
                         AND DEPARTMENTS.LOCATION_ID =
LOCATIONS.LOCATION_ID
                         AND LOCATIONS.COUNTRY_ID =
COUNTRIES.COUNTRY_ID) Q1,
(SELECT DISTINCT JOB_ID, COUNT (EMPLOYEE_ID) EMPL_CONTRATADOS
                      FROM HR.EMPLOYEES
                     WHERE SUBSTR (TO_CHAR (HIRE_DATE,
'yyyy/mm/dd'), 1, 4) BETWEEN
(SUBSTR(TO_CHAR(SYSDATE,'yyyy/mm/dd'),1,4)- 5)
AND SUBSTR (TO_CHAR (SYSDATE,'yyyy/mm/dd'),1,4)
                  GROUP BY JOB_ID) Q2,
(SELECT DISTINCT JOB_ID, COUNT (EMPLOYEE_ID) TOTAL_JOBID
FROM HR.EMPLOYEES GROUP BY JOB_ID) Q3
WHERE Q1.JOB_ID = Q2.JOB_ID AND Q1.JOB_ID = Q3.JOB_ID
ORDER BY EMPL_CONTRATADOS DESC) INLINEVIEW_1
WHERE ROWNUM <= 10;
```

- Mejora c3_p2_toad:

```
SELECT /*c3_p2_toad*/ JOB_ID,
       DEPARTMENT_ID,
       COUNTRY_NAME,
       EMPL_CONTRATADOS,
       TOTAL_JOBID
  FROM (  SELECT Q1.JOB_ID,
                 DEPARTMENT_ID,
                 COUNTRY_NAME,
                 EMPL_CONTRATADOS,
                 TOTAL_JOBID
            FROM (SELECT DISTINCT JOB_ID, EMPLOYEES.DEPARTMENT_ID,
COUNTRY_NAME
                    FROM HR.EMPLOYEES
                         INNER JOIN HR.DEPARTMENTS
                            ON (EMPLOYEES.DEPARTMENT_ID =
                                   DEPARTMENTS.DEPARTMENT_ID)
                         INNER JOIN HR.LOCATIONS
                            ON (DEPARTMENTS.LOCATION_ID =
LOCATIONS.LOCATION_ID)
                         INNER JOIN HR.COUNTRIES
```

```
                            ON (LOCATIONS.COUNTRY_ID =
COUNTRIES.COUNTRY_ID))
                  Q1
                  INNER JOIN
                  (  SELECT DISTINCT JOB_ID, COUNT (EMPLOYEE_ID)
EMPL_CONTRATADOS
                       FROM HR.EMPLOYEES
                      WHERE SUBSTR (TO_CHAR (HIRE_DATE, 'yyyy/mm/dd'),
1, 4) BETWEEN (SUBSTR(TO_CHAR(SYSDATE,'yyyy/mm/dd'),1,4)- 5)

AND SUBSTR (TO_CHAR(SYSDATE,'yyyy/mm/dd'),1,4)
                   GROUP BY JOB_ID) Q2
                     ON (Q1.JOB_ID = Q2.JOB_ID)
                  INNER JOIN
                  (SELECT DISTINCT JOB_ID, COUNT (EMPLOYEE_ID)
TOTAL_JOBID
                       FROM HR.EMPLOYEES
                   GROUP BY JOB_ID) Q3
                     ON (Q1.JOB_ID = Q3.JOB_ID)
        ORDER BY EMPL_CONTRATADOS DESC) INLINEVIEW_1
 WHERE ROWNUM <= 10;
```

- Mejora c3_p3a_toad y mejora c3_p3b_toad:

```
select job_id,
       department_id,
       country_name,
       Empl_Contratados,
       Total_JobId
  from (select q1.job_id,
               q1.department_id,
               q1.country_name,
               q2.Empl_Contratados,
               q3.Total_JobId
          from (select /*+ FULL(DEPARTMENTS) */ distinct
EMPLOYEES1.job_id,
                       EMPLOYEES1.department_id,
                       countries.country_name
                  from hr.employees EMPLOYEES1,
                       hr.departments,
                       hr.locations,
                       hr.countries
                 where EMPLOYEES1.department_id =
departments.department_id
                   and departments.location_id = locations.location_id
                   and locations.country_id = countries.country_id)
q1,
               (select distinct job_id,
                       count(employee_id) as Empl_Contratados
                  from hr.employees EMPLOYEES2
                 where substr(TO_CHAR(hire_date, 'yyyy/mm/dd'), 1, 4)
between substr(TO_CHAR(SYSDATE, 'yyyy/mm/dd'), 1, 4) - 5 and
substr(TO_CHAR(SYSDATE, 'yyyy/mm/dd'), 1, 4)
                 group by job_id) q2,
               (select distinct job_id,
                       count(employee_id) as Total_JobId
                  from hr.employees EMPLOYEES3
                 group by job_id) q3
         where q1.job_id = q2.job_id
           and q1.job_id = q3.job_id
         order by q2.Empl_Contratados desc) V
```

```
 where rownum <= 10;
```

- Mejora c3_p5a_toad y mejora c3_p5b_toad:

```
select job_id,
       department_id,
       country_name,
       Empl_Contratados,
       Total_JobId
  from (select /*+ PX_JOIN_FILTER(Q1) */ q1.job_id,
               q1.department_id,
               q1.country_name,
               q2.Empl_Contratados,
               q3.Total_JobId
          from (select distinct EMPLOYEES1.job_id,
                       EMPLOYEES1.department_id,
                       countries.country_name
                  from hr.employees EMPLOYEES1,
                       hr.departments,
                       hr.locations,
                       hr.countries
                 where EMPLOYEES1.department_id =
departments.department_id
                   and departments.location_id = locations.location_id
                   and locations.country_id = countries.country_id)
q1,
               (select distinct job_id,
                       count(employee_id) as Empl_Contratados
                  from hr.employees EMPLOYEES2
                 where substr(TO_CHAR(hire_date, 'yyyy/mm/dd'), 1, 4)
between substr(TO_CHAR(SYSDATE, 'yyyy/mm/dd'), 1, 4) - 5 and
substr(TO_CHAR(SYSDATE, 'yyyy/mm/dd'), 1, 4)
                 group by job_id) q2,
               (select distinct job_id,
                       count(employee_id) as Total_JobId
                  from hr.employees EMPLOYEES3
                 group by job_id) q3
         where q1.job_id = q2.job_id
           and q1.job_id = q3.job_id
         order by q2.Empl_Contratados desc) V
 where rownum <= 10;
```

- Mejora c3_p6a_toad:

```
SELECT job_id,
       department_id,
       country_name,
       Empl_Contratados,
       Total_JobId
  FROM (SELECT q1.job_id,
               q1.department_id,
               q1.country_name,
               q2.Empl_Contratados,
               q3.Total_JobId
          FROM (SELECT DISTINCT EMPLOYEES1.job_id,
                       EMPLOYEES1.department_id,
                       countries.country_name
                  FROM hr.locations,
                       hr.departments,
                       hr.countries,
                       hr.employees EMPLOYEES1
```

```
                WHERE EMPLOYEES1.department_id =
departments.department_id + 0
                  AND departments.location_id = locations.location_id
+ 0
                  AND countries.country_id = locations.country_id ||
'') q1,
               (SELECT DISTINCT job_id,
                       COUNT(employee_id) AS Empl_Contratados
                  FROM hr.employees EMPLOYEES2
                 WHERE SUBSTR(TO_CHAR(hire_date, 'yyyy/mm/dd'), 1, 4)
BETWEEN SUBSTR(TO_CHAR(SYSDATE, 'yyyy/mm/dd'), 1, 4) - 5 AND
SUBSTR(TO_CHAR(SYSDATE, 'yyyy/mm/dd'), 1, 4)
                 GROUP BY job_id) q2,
               (SELECT DISTINCT job_id,
                       COUNT(employee_id) AS Total_JobId
                  FROM hr.employees EMPLOYEES3
                 GROUP BY job_id) q3
         WHERE q2.job_id = q1.job_id
           AND q3.job_id = q1.job_id
           AND q3.job_id = q2.job_id
         ORDER BY q2.Empl_Contratados DESC) V
 WHERE 10 >= ROWNUM
```

- Mejora c3_p6b_toad:

```
/*c3_p6b_toad; Alternativa 24 es la que mejor equilibra coste, tiempo
y CPU*/
SELECT job_id,
       department_id,
       country_name,
       Empl_Contratados,
       Total_JobId
  FROM (SELECT /*+ NO_USE_HASH(Q3,Q2,Q1) */ q1.job_id,
               q1.department_id,
               q1.country_name,
               q2.Empl_Contratados,
               q3.Total_JobId
          FROM (SELECT DISTINCT EMPLOYEES1.job_id,
                       EMPLOYEES1.department_id,
                       countries.country_name
                  FROM hr.employees EMPLOYEES1,
                       hr.departments,
                       hr.locations,
                       hr.countries
                 WHERE            EMPLOYEES1.department_id            =
departments.department_id
                   AND departments.location_id = locations.location_id
                   AND  locations.country_id  =  countries.country_id)
q1,
               (SELECT DISTINCT job_id,
                       COUNT(employee_id) AS Empl_Contratados
                  FROM hr.employees EMPLOYEES2
                 WHERE  SUBSTR(TO_CHAR(hire_date,  'yyyy/mm/dd'),  1,  4)
BETWEEN   SUBSTR(TO_CHAR(SYSDATE,   'yyyy/mm/dd'),   1,   4)   -   5   AND
SUBSTR(TO_CHAR(SYSDATE, 'yyyy/mm/dd'), 1, 4)
                 GROUP BY job_id) q2,
               (SELECT DISTINCT job_id,
                       COUNT(employee_id) AS Total_JobId
                  FROM hr.employees EMPLOYEES3
                 GROUP BY job_id) q3
         WHERE q1.job_id = q2.job_id
```

```
          AND q1.job_id = q3.job_id
        ORDER BY q2.Empl_Contratados DESC) V
 WHERE ROWNUM <= 10
```

- Mejora c3_p7a_toad:

```
select job_id,
       department_id,
       country_name,
       Empl_Contratados,
       Total_JobId
  from (select q1.job_id,
               q1.department_id,
               q1.country_name,
               q2.Empl_Contratados,
               q3.Total_JobId
          from (select /*+ ORDERED */ distinct EMPLOYEES1.job_id,
                       EMPLOYEES1.department_id,
                       countries.country_name
                  from hr.employees EMPLOYEES1,
                       hr.departments,
                       hr.locations,
                       hr.countries
                 where EMPLOYEES1.department_id =
departments.department_id
                   and departments.location_id = locations.location_id
                   and locations.country_id = countries.country_id)
q1,
               (select distinct job_id,
                       count(employee_id) as Empl_Contratados
                  from hr.employees EMPLOYEES2
                 where substr(TO_CHAR(hire_date, 'yyyy/mm/dd'), 1, 4)
between substr(TO_CHAR(SYSDATE, 'yyyy/mm/dd'), 1, 4) - 5 and
substr(TO_CHAR(SYSDATE, 'yyyy/mm/dd'), 1, 4)
                 group by job_id) q2,
               (select distinct job_id,
                       count(employee_id) as Total_JobId
                  from hr.employees EMPLOYEES3
                 group by job_id) q3
         where q1.job_id = q2.job_id
           and q1.job_id = q3.job_id
         order by q2.Empl_Contratados desc) V
 where rownum <= 10;
```

- Mejora c3_p7b_toad:

```
select job_id,
       department_id,
       country_name,
       Empl_Contratados,
       Total_JobId
  from (select q1.job_id,
               q1.department_id,
               q1.country_name,
               q2.Empl_Contratados,
               q3.Total_JobId
          from (select distinct EMPLOYEES1.job_id,
                       EMPLOYEES1.department_id,
                       countries.country_name
                  FROM hr.locations,
                       hr.departments,
```

```
                        hr.countries,
                        hr.employees EMPLOYEES1
                  where EMPLOYEES1.department_id =
departments.department_id + 0
                    AND departments.location_id = locations.location_id
+ 0
                    AND countries.country_id = locations.country_id ||
'') q1,
                 (select distinct job_id,
                        count(employee_id) as Empl_Contratados
                    from hr.employees EMPLOYEES2
                  where substr(TO_CHAR(hire_date, 'yyyy/mm/dd'), 1, 4)
between substr(TO_CHAR(SYSDATE, 'yyyy/mm/dd'), 1, 4) - 5 and
substr(TO_CHAR(SYSDATE, 'yyyy/mm/dd'), 1, 4)
                  group by job_id) q2,
                 (select distinct job_id,
                        count(employee_id) as Total_JobId
                    from hr.employees EMPLOYEES3
                  group by job_id) q3
          where q2.job_id = q1.job_id
            AND q3.job_id = q1.job_id
            AND q3.job_id = q2.job_id
          order by q2.Empl_Contratados desc) V
 where 10 >= rownum - UID * 0;
```

- Mejora c3_p8a_toad:

```
select job_id,
       department_id,
       country_name,
       Empl_Contratados,
       Total_JobId
  from (select /*+ USE_HASH(Q3,Q2) ORDERED */ q1.job_id,
               q1.department_id,
               q1.country_name,
               q2.Empl_Contratados,
               q3.Total_JobId
          from (select distinct EMPLOYEES1.job_id,
                        EMPLOYEES1.department_id,
                        countries.country_name
                   FROM hr.departments,
                        hr.countries,
                        hr.locations,
                        hr.employees EMPLOYEES1
                  where EMPLOYEES1.department_id =
departments.department_id + 0
                    and locations.location_id = departments.location_id
+ 0
                    and locations.country_id = countries.country_id ||
'') q1,
                 (select distinct job_id,
                        count(employee_id) as Empl_Contratados
                    from hr.employees EMPLOYEES2
                  where substr(TO_CHAR(hire_date, 'yyyy/mm/dd'), 1, 4)
between substr(TO_CHAR(SYSDATE, 'yyyy/mm/dd'), 1, 4) - 5 and
substr(TO_CHAR(SYSDATE, 'yyyy/mm/dd'), 1, 4)
                  group by job_id) q2,
                 (select distinct job_id,
                        count(employee_id) as Total_JobId
                    from hr.employees EMPLOYEES3
                  group by job_id) q3
```

```
         where q1.job_id = q2.job_id
           and q1.job_id = q3.job_id
         order by q2.Empl_Contratados desc) V
 where rownum <= 10;
```

- Mejora c3_p8b_toad:

```
select /*+ FIRST_ROWS(30) */ job_id,
       department_id,
       country_name,
       Empl_Contratados,
       Total_JobId
  from (select q1.job_id,
               q1.department_id,
               q1.country_name,
               q2.Empl_Contratados,
               q3.Total_JobId
          from (select distinct EMPLOYEES1.job_id,
                       EMPLOYEES1.department_id,
                       countries.country_name
                  FROM hr.locations,
                       hr.departments,
                       hr.countries,
                       hr.employees EMPLOYEES1
                 where EMPLOYEES1.department_id =
departments.department_id + 0
                   AND departments.location_id = locations.location_id
+ 0
                   AND countries.country_id = locations.country_id ||
'') q1,
               (select distinct job_id,
                       count(employee_id) as Empl_Contratados
                  from hr.employees EMPLOYEES2
                 where substr(TO_CHAR(hire_date, 'yyyy/mm/dd'), 1, 4)
between substr(TO_CHAR(SYSDATE, 'yyyy/mm/dd'), 1, 4) - 5 and
substr(TO_CHAR(SYSDATE, 'yyyy/mm/dd'), 1, 4)
                 group by job_id) q2,
               (select distinct job_id,
                       count(employee_id) as Total_JobId
                  from hr.employees EMPLOYEES3
                 group by job_id) q3
         where q2.job_id = q1.job_id
           AND q3.job_id = q1.job_id
           AND q3.job_id = q2.job_id
         order by q2.Empl_Contratados desc) V
 where 10 >= rownum - UID * 0;
```

9.4 Propuesta de guía de trabajo en TOAD

Este guión tiene como objetivo mostrar paso a paso el proceso de optimización de una consulta hipotética empleando los mecanismos provistos por TOAD. Al tratarse de una prueba gratuita de 30 días, se recomienda haber trabajado previamente con SQL Developer y conocer qué tipo de información debe ser extraída para valorar la mejora o deterioro de la sentencia (un esquema de los pasos a dar ha sido incluido al final del anexo 7.2).

Suponiendo este conocimiento previo, se empleará como modelo el segundo caso de prueba, la consulta 2. Los principales parámetros en los que se basarán las decisiones son el tiempo transcurrido, el coste del plan, las medidas estimadas de tuplas extraíbles y tiempo de ejecución, y el número real de tuplas resultado, así como el número de lecturas lógicas y físicas. Al tratarse de una consulta, el número de escrituras no es relevante, pero debería ser tenido en cuenta si la sentencia SQL que se desea optimizar incluye operaciones de escritura, como CREATE, INSERT o UPDATE.

En primer lugar deben definirse el conjunto de pruebas que se realizarán para buscar potenciales versiones optimizadas de la consulta original. Para ello es recomendable asignar un identificador a cada prueba y una descripción de la misma indicando qué cambios en la configuración se vayan a realizar, si los hubiera. Este guión seguirá la misma nomenclatura que el estudio: cX_pY, siendo X el número de la consulta que se esté analizando (si hubiera más de una) e Y el número de prueba.

A continuación, se conecta en primer lugar a la base de datos. Para ello puede reutilizarse una de las conexiones anteriormente añadidas o crear una nueva (en caso de duda, consultar anexo 7.1). Aparecerá el siguiente escenario, en el que el editor ocupa la parte central. En él debe escribirse la consulta, cuidando de hacer referencia al esquema en cada tabla empleada, como hr.employees:

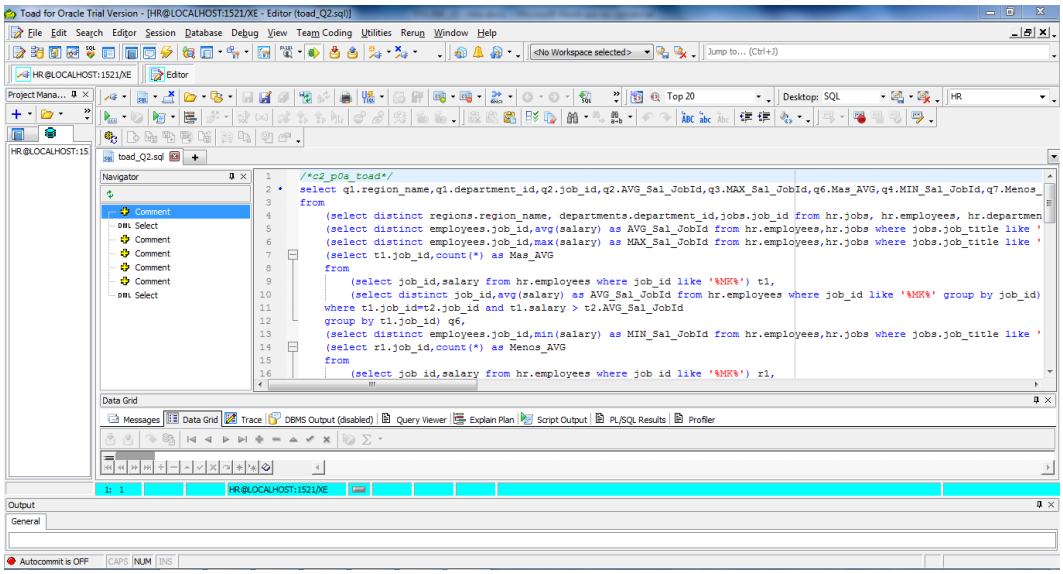


Ilustración 50: Guión TOAD (I)

El panel del navegador por defecto no muestra los comentarios, pero es recomendable activarlo para poder tener un mejor seguimiento del documento. Como buenas prácticas, los comentarios deben ser breves pero explicativos y por ello se han empleado los identificadores de las pruebas (aparecen en verde).

Lo siguiente es configurar las opciones para extraer la información del rendimiento de la ejecución. Para ello, pulsar el botón derecho, y seleccionar las opciones 'Auto Trace' y 'SQL Trace (tkprof) > Enable - Retrieve trace file after execution':

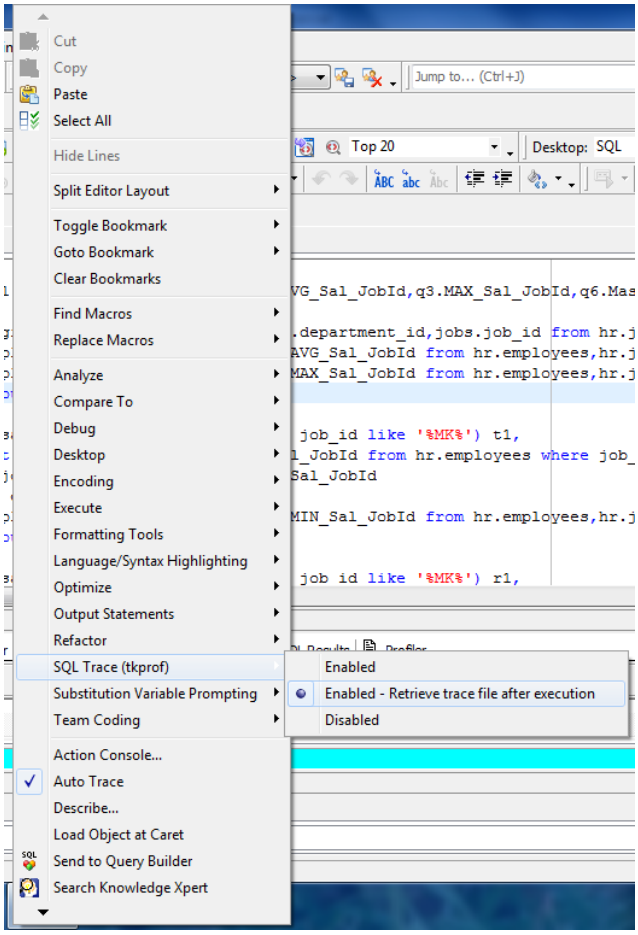


Ilustración 51: Guión TOAD (II)

Para establecer un punto de partida frente al cual comparar los sucesivos resultados, existen dos posibilidades: la primera es ejecutar la sentencia dos veces (para en la segunda contar con que la información ha sido traída a memoria) o comenzar a mejorar directamente la consulta porque el optimizador incluye en su búsqueda de alternativas la ejecución de la primera consulta (y la ejecuta también dos veces). En este caso se opta por la primera vía para presentar el panel de navegación del trace file.

Se selecciona la sentencia (en este caso consulta) que se desea ejecutar y se selecciona la opción "Execute / Compile Statement". Tras su ejecución aparecerá una nueva pestaña llamada "Trace file browser", que contiene la información de la ejecución. Es importante cerrar este navegador después de cada ejecución para que la

siguiente pueda emplear el mismo tracefile, de otra manera la ejecución se llevaría a cabo sin mostrar las estadísticas del tracefile actualizadas:

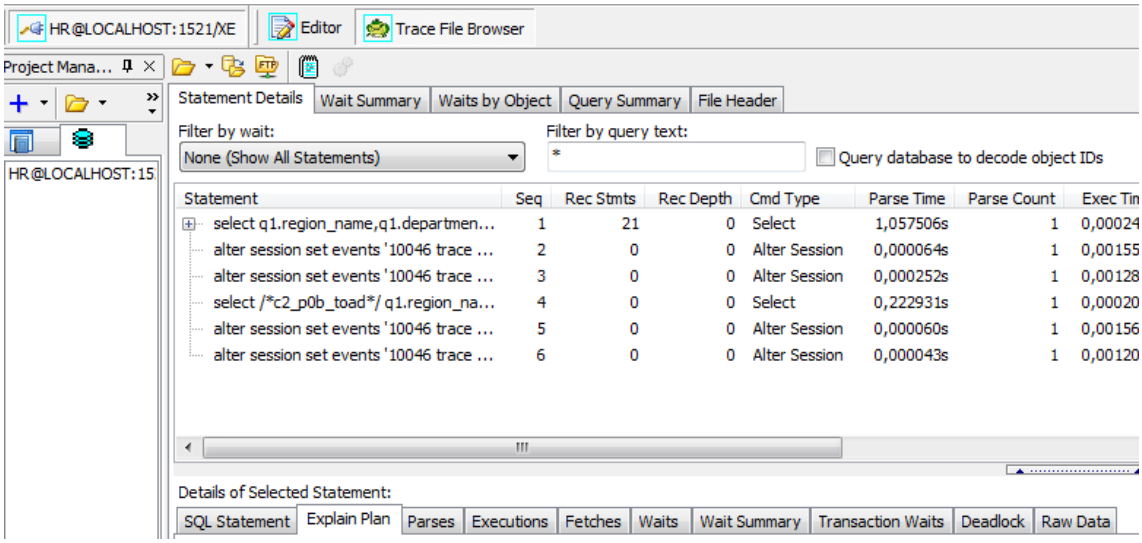


Ilustración 52: Guión TOAD (III)

Para extraer el explain plan basta con ir a "Editor" > "Explain Plan Current SQL", y en el panel inferior al editor aparecerá. Puede escogerse además el modo en que se muestra el explain plan, siendo por defecto 'Tree'. Es necesario tener en cuenta que este es el explain plan, no el plan de ejecución, por lo que las medidas son todas *estimadas* (en caso de seleccionar display mode = dbms_xplan, se vería con más claridad). Las estadísticas de lo ocurrido aparecerán en el trace file:

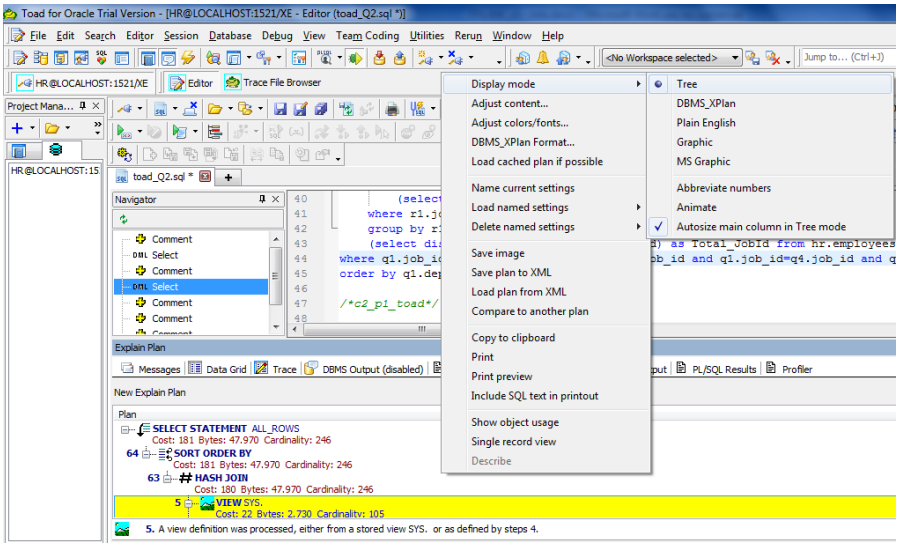


Ilustración 53: Guión TOAD (IV)

Una vez ejecutada la consulta y preparada la configuración, es el momento de buscar posibles mejoras. Para ello, existen dos posibilidades, "Auto Optimize SQL" (en adelante AO SQL) y "Advanced SQL Optimization" (Adv SQL Opt), ya que la tercera (OEM, Oracle Tuning Advisor) requiere de licencia aparte. Se ejecutará una prueba con AO SQL y otra con Adv SQL Opt.

- AO SQL: Sobre la sentencia que se quiere auto-optimizar, se selecciona, se pulsa el botón derecho y se indica esta opción. A continuación abre una nueva pestaña con para el navegador "Auto Optimize SQL", contigua al editor:

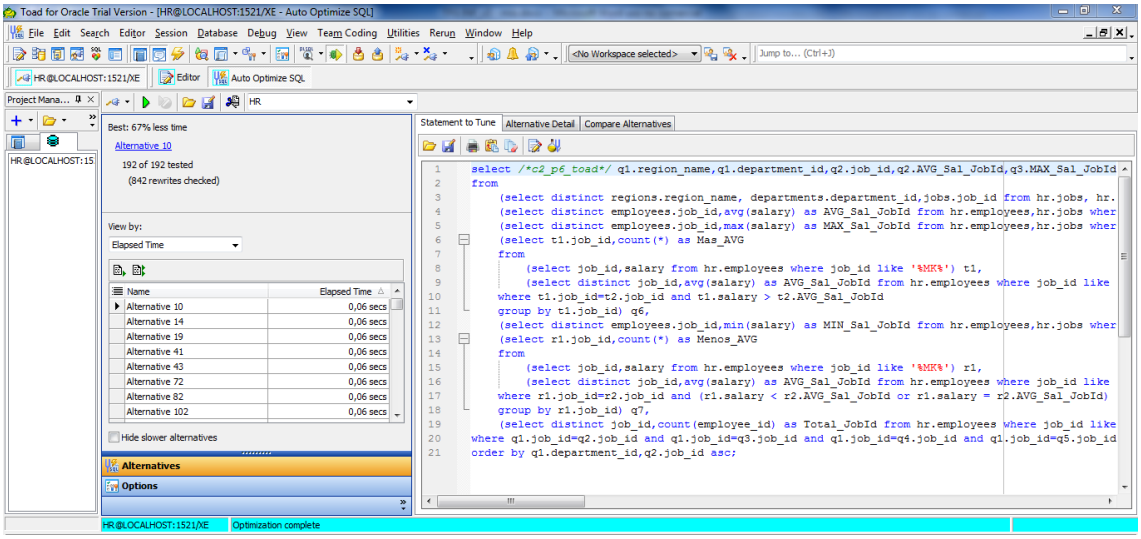


Ilustración 54: Guión TOAD (V)

Dentro de ella se distinguen varias secciones: 1) la sentencia que se desea optimizar, 2) detalle de la alternativa (en este caso sólo se puede ver una alternativa, no todas como en Adv SQL Opt), y 3) Comparar alternativas, que compara la original con la alternativa que se seleccione. El panel de la izquierda muestra cuántos escenarios se han considerado (842), cuántos de ellos se han seleccionado y están disponibles como alternativas (192), el criterio de ordenación de las alternativas y los identificadores de cada una de ellas junto con el valor del parámetro de ordenación.

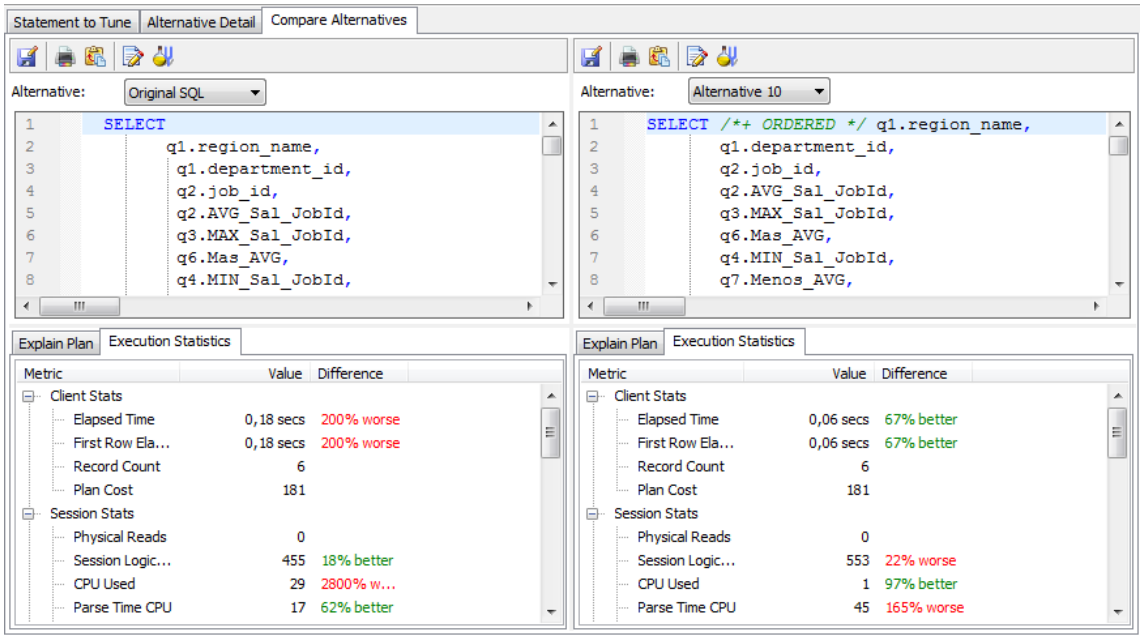


Ilustración 55: Guión TOAD (VI)

- Adv SQL Opt:

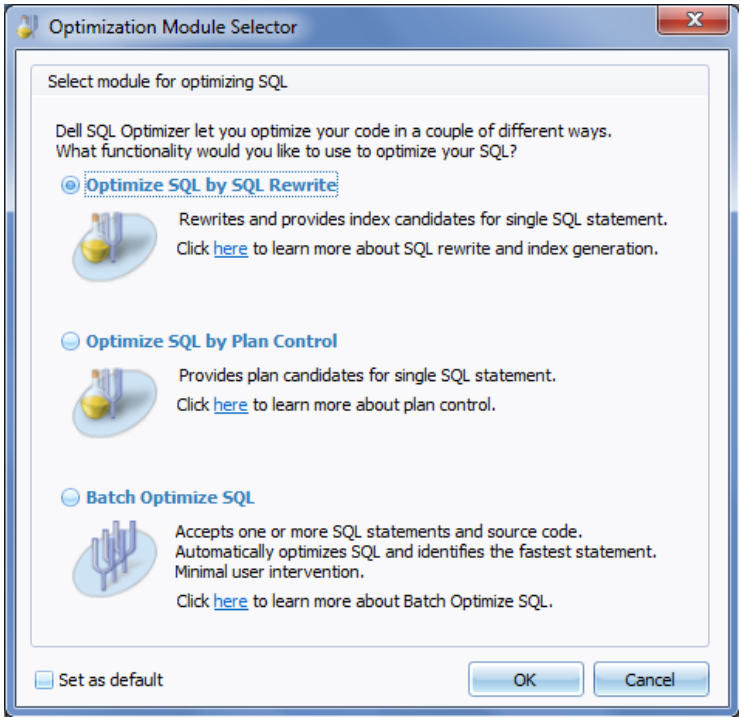


Ilustración 56: Guión TOAD (VII)

Ilustración 57: Adv SQL Opt (I)

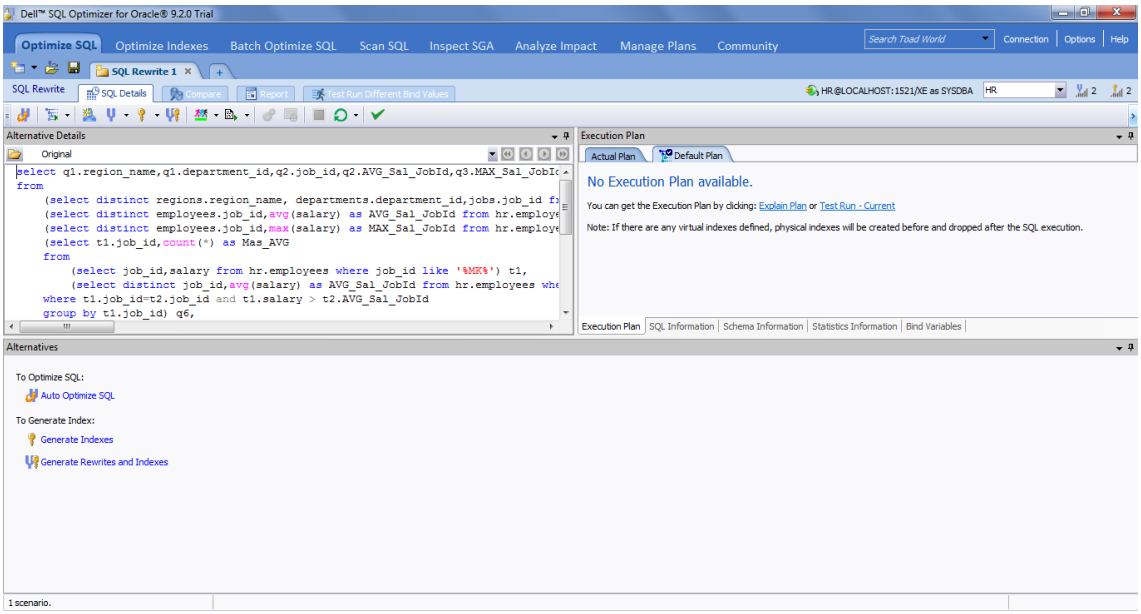


Ilustración 58: Guión TOAD (VIII)

Ilustración 59: Adv SQL Opt (II)

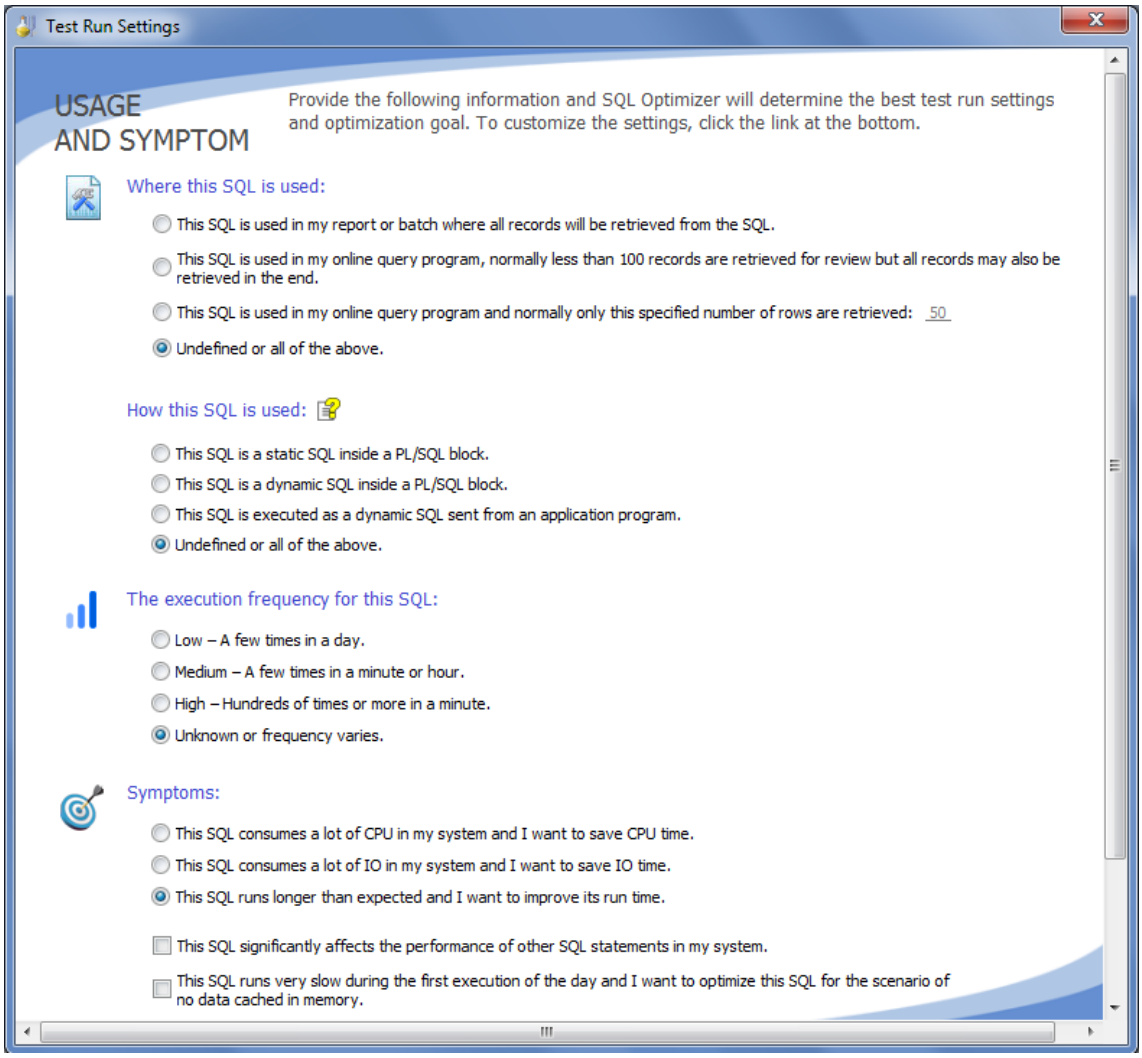


Ilustración 60: Guión TOAD (IX)

Ilustración 61: Adv SQL Opt (III)

> Start test run. Comienza a buscar escenarios (192), de los cuales un subgrupo (142 en este caso) pasará a formar las potenciales alternativas a la consulta original:

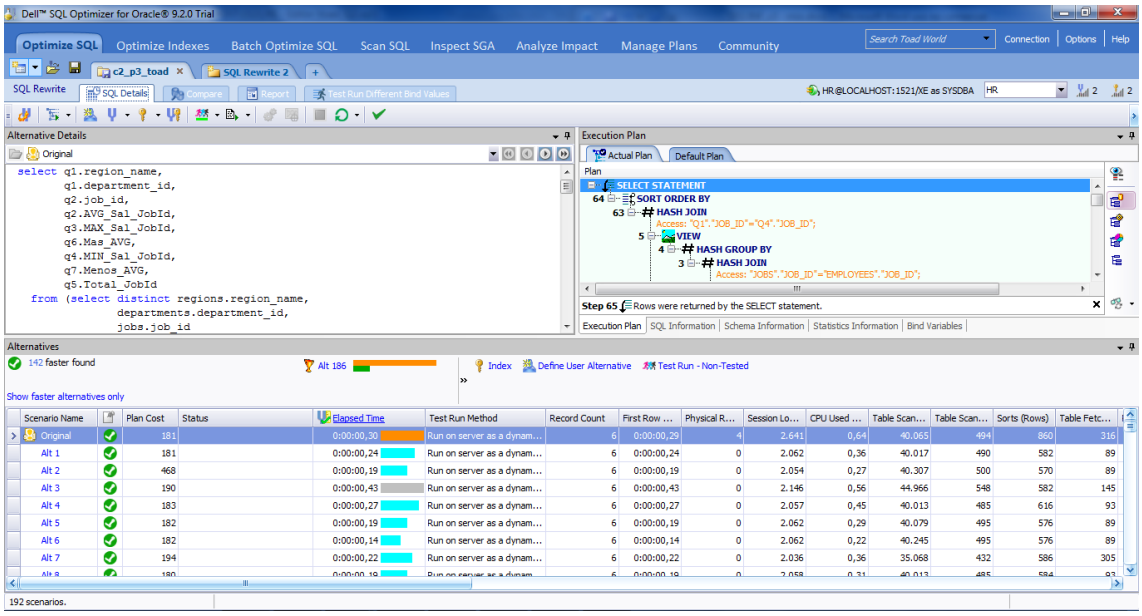


Ilustración 62: Guión TOAD (X)

En la pestaña "Compare" se pueden ver las estadísticas de todas las alternativas y además comparar los resultados específicos de la original respecto a una alternativa (la que esté por encima de la línea naranja). Desde aquí se da la opción al usuario de enviar el texto de la sentencia alternativa al editor cuando se considera que es una versión optimizada.

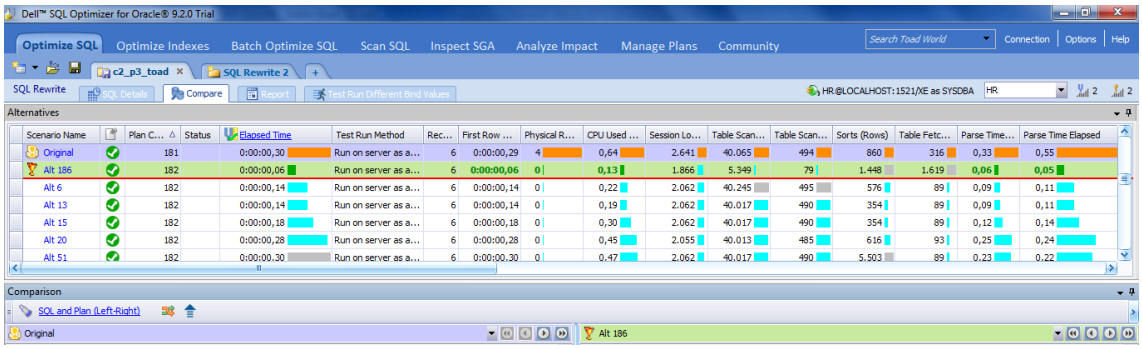


Ilustración 63: Guión TOAD (XI)

Si el criterio de ordenación de las alternativas es otro distinto al tiempo transcurrido ("time elapsed"), pulsando sobre cualquiera de las columnas se ordenarán los escenarios generados en función de esta. Por ejemplo, si se cambiara a coste del plan, se obtendría otra alternativa:

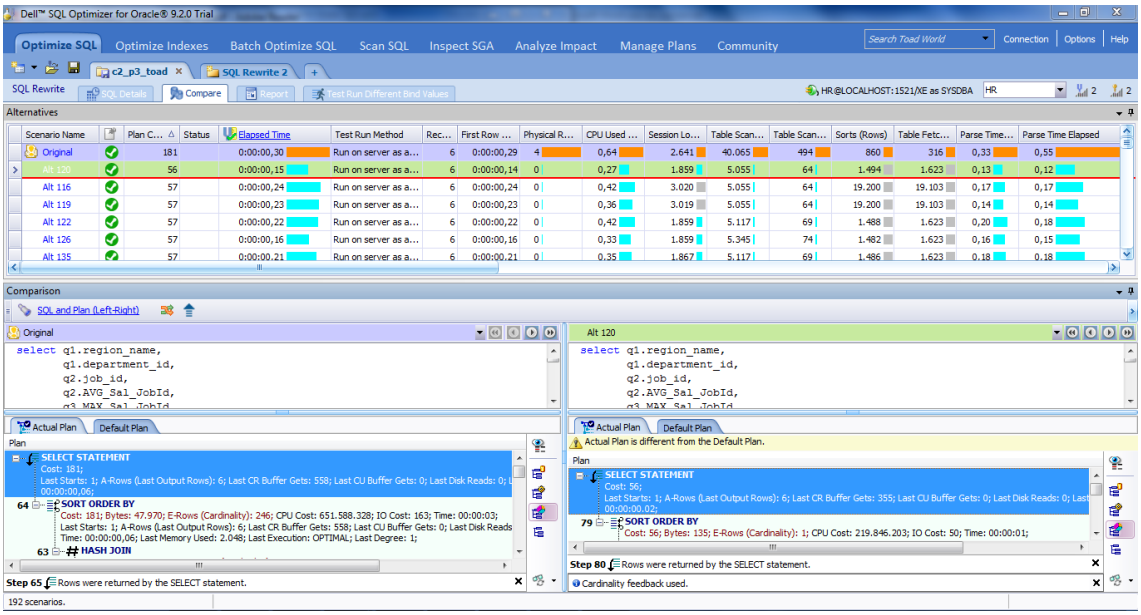


Ilustración 64: Guión TOAD (XII)

En ocasiones varios escenarios coinciden en el valor del parámetro utilizado como primer criterio, por lo que en este caso es recomendable aplicar un segundo criterio que ayude a discriminar las opciones menos óptimas, y continuar este proceso hasta llegar sólo a una.

Al finalizar todas las pruebas inicialmente planeadas, se obtiene un reducido grupo de alternativas, las mejores de las mejores (recordar que ya el optimizador hace una primera criba al no incluir como alternativas todos los escenarios generados), entre las cuales deberá escogerse la que mejor satisfaga las necesidades del usuario. Esto dependerá del tipo de entorno en que se vaya a ejecutar la consulta.

9.5 Script TOAD - Privilegios sobre la base de datos

En este anexo se incluye el script oficial facilitado por TOAD en su manual integrado para la concesión de privilegios sobre la base de datos:

```
Database Privileges Script

The following script provides the commands required to grant the Oracle
database privileges needed for various functions in SQL Optimizer.
/* This is a sample script for granting all the privileges required by the
SQL Optimizer user. */
/* In order to grant privileges on SYS objects, you must login with the SYS
user */
/* Please verify the script below before executing. */

/* Create a new role */
create role sqloptimizer_user;

/* For the Run on Server function */
grant EXECUTE on SYS.DBMS_SQL to sqloptimizer_user;

/* For retrieving run time statistics */
grant SELECT on SYS.V_$MYSTAT to sqloptimizer_user;
grant SELECT on SYS.V_$STATNAME to sqloptimizer_user;

/* For retrieving trace statistics */
grant SELECT on SYS.V_$PARAMETER to sqloptimizer_user;

/* For trace setup */
grant SELECT on SYS.V_$SESSION to sqloptimizer_user;
grant SELECT on SYS.V_$PROCESS to sqloptimizer_user;

/* For retrieving actual plan */
grant SELECT on SYS.V_$SQLAREA to sqloptimizer_user;
grant SELECT on SYS.V_$ SQL_PLAN_STATISTICS_ALL to sqloptimizer_user;
grant SELECT on SYS.V_$SESSION to sqloptimizer_user;
grant EXECUTE on SYS.DBMS_XPLAN to sqloptimizer_user;
grant ALTER SESSION to sqloptimizer_user;

/* For capturing bind values from database */
grant SELECT on SYS.V_$SQLAREA to sqloptimizer_user;
grant SELECT on SYS.V_$SQL_BIND_CAPTURE to sqloptimizer_user;

/* For altering session parameters */
grant SELECT on SYS.V_$PARAMETER to sqloptimizer_user;

/* For deploying stored outlines */
grant CREATE ANY OUTLINE to sqloptimizer_user;
grant DROP ANY OUTLINE to sqloptimizer_user;
grant SELECT, UPDATE on OUTLN.OL$HINTS to sqloptimizer_user;
grant SELECT, UPDATE on OUTLN.OL$ to sqloptimizer_user;
grant SELECT, UPDATE on OUTLN.OL$NODES to sqloptimizer_user;

/* For generating plans in Plan Control session */
grant ADMINISTER SQL MANAGEMENT OBJECT to sqloptimizer_user;
grant EXECUTE on SYS.DBMS_SPM to sqloptimizer_user;
grant EXECUTE on SYS.DBMS_XPLAN to sqloptimizer_user;
grant SELECT on SYS.DBA_SQL_PLAN_BASELINES to sqloptimizer_user;
```

```
grant SELECT on SYS.V_$SQLAREA to sqloptimizer_user;
grant SELECT on SYS.V_$SQLTEXT_WITH_NEWLINES to sqloptimizer_user;

/* For inspecting executed SQL from SGA */
grant SELECT on SYS.V_$SQLAREA to sqloptimizer_user;
grant SELECT on SYS.V_$SQLTEXT_WITH_NEWLINES to sqloptimizer_user;

/* For inspecting currently executing SQL from SGA */
grant SELECT on SYS.V_$SQLAREA to sqloptimizer_user;
grant SELECT on SYS.V_$SQLTEXT_WITH_NEWLINES to sqloptimizer_user;
grant SELECT on SYS.V_$OPEN_CURSOR to sqloptimizer_user;
grant SELECT on SYS.V_$SESSION to sqloptimizer_user;

/* For inspecting currently executing SQL from SGA and monitor by session*/
grant SELECT on SYS.V_$SESSION to sqloptimizer_user;

/*For retrieving the plan information (applicable to Oracle 9 or later) */
grant SELECT on SYS.V_$SQL_PLAN to sqloptimizer_user;

/* For flushing the shared pool */
grant ALTER SYSTEM to sqloptimizer_user;

/* For viewing baselines and their plans */
grant SELECT on SYS.DBA_SQL_PLAN_BASELINES to sqloptimizer_user;
grant EXECUTE on SYS.DBMS_XPLAN to sqloptimizer_user;

/* For exporting, importing and migrating baselines */
grant ADMINISTER SQL MANAGEMENT OBJECT to sqloptimizer_user;
grant CREATE TABLE to sqloptimizer_user;
grant EXECUTE on SYS.DBMS_SPM to sqloptimizer_user;
grant SELECT on SYS.DBA_SQL_PLAN_BASELINES to sqloptimizer_user;

/* For viewing and modifying baseline configuration values */
grant ADMINISTER SQL MANAGEMENT OBJECT to sqloptimizer_user;
grant ALTER SYSTEM to sqloptimizer_user;
grant EXECUTE on SYS.DBMS_SPM to sqloptimizer_user;
grant SELECT on SYS.V_$SYSTEM_PARAMETER to sqloptimizer_user;
grant SELECT on SYS.DBA_SQL_MANAGEMENT_CONFIG to sqloptimizer_user;
grant SELECT on SYS.DBA_DATA_FILES to sqloptimizer_user;

/* For managing stored outlines */
grant DROP ANY OUTLINE to sqloptimizer_user;
grant ALTER ANY OUTLINE to sqloptimizer_user;
grant EXECUTE on SYS.OUTLN_PKG to sqloptimizer_user;
grant SELECT on OUTLN.OL$HINTS to sqloptimizer_user;
grant SELECT on OUTLN.OL$ to sqloptimizer_user;

/* For recommending indexes in Optimize Indexes */
grant SELECT on SYS.V_$SESSION to sqloptimizer_user;

/* For collecting SQL from AWR in Optimize Indexes */
grant SELECT on SYS.DBA_HIST_SNAPSHOT to sqloptimizer_user;
grant SELECT on SYS.DBA_HIST_SQLTEXT to sqloptimizer_user;
grant SELECT on SYS.DBA_HIST_SQLSTAT to sqloptimizer_user;

/* For Optimize Indexes, Analyze Impact displaying control information from
AWR */
grant SELECT on SYS.DBA_HIST_WR_CONTROL to sqloptimizer_user;
```

```
/* For Optimize Indexes, Analyze Impact displaying SQL summary from AWR */
grant SELECT on SYS.DBA_HIST_SQL_SUMMARY to sqloptimizer_user;

/* For collecting SQL from Foglight PA Repository in Optimize Indexes */
grant SELECT on QUEST_SC_ACTION_DIM to sqloptimizer_user;
grant SELECT on QUEST_SC_CLIENT_INFO_DIM to sqloptimizer_user;
grant SELECT on QUEST_SC_MODULE_DIM to sqloptimizer_user;
grant SELECT on QUEST_SC_SQL_STAT_FACT to sqloptimizer_user;
grant SELECT on QUEST_SC_SQL_SYNTAX_DIM to sqloptimizer_user;
grant SELECT on QUEST_CTRL_PYRAMID_LEVELS to sqloptimizer_user;
grant SELECT on QUEST_DB_USER_DIM to sqloptimizer_user;
grant SELECT on QUEST_INSTANCE_DIM to sqloptimizer_user;
grant SELECT on QUEST_PROGRAM_DIM to sqloptimizer_user;
grant SELECT on QUEST_TIME_DIM to sqloptimizer_user;

/* For collecting SQL from SGA in Optimize Indexes */
grant SELECT on SYS.V_$SQLAREA to sqloptimizer_user;

/* For Analyze Impact index impact */
grant SELECT on SYS.V_$SESSION to sqloptimizer_user;

/* For Analyze Impact index impact, parameter impact */
grant SELECT on SYS.V_$PARAMETER to sqloptimizer_user;

/* For Analyze Impact compare database impact */
grant SELECT on SYS.PRODUCT_COMPONENTS_VERSION to sqloptimizer_user;
grant SELECT on SYS.V_$PARAMETER to sqloptimizer_user;
grant SELECT on SYS.V_$SGAINFO to sqloptimizer_user;
grant SELECT on SYS.V_$DATABASE to sqloptimizer_user;
grant SELECT on SYS.V_$INSTANCE to sqloptimizer_user;
grant SELECT on SYS.V_$STATISTICS_LEVEL to sqloptimizer_user;
grant SELECT on SYS.V_$OPTION to sqloptimizer_user;
grant SELECT on SYS.NLS_SESSION_PARAMETERS to sqloptimizer_user;
grant SELECT on SYS.NLS_INSTANCE_PARAMETERS to sqloptimizer_user;
grant SELECT on SYS.NLS_DATABASE_PARAMETERS to sqloptimizer_user;

/* Grant to the role: SELECT_CATALOG_ROLE */
grant SELECT_CATALOG_ROLE to sqloptimizer_user;

/* For retrieving DBMS_XPLAN */
grant EXECUTE on SYS.DBMS_XPLAN to sqloptimizer_user;

/* Grant the role to a user */
grant sqloptimizer_user to &username;

/* For create a SQL Translation Profile to use in the SQL Translation window
*/
grant CREATE SQL TRANSLATION PROFILE to sqloptimizer_user;
```